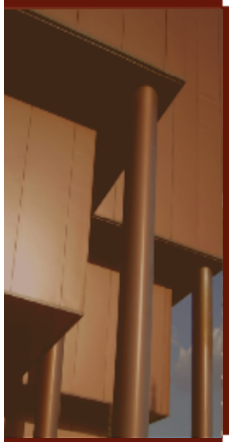
Proyecto Fin de Carrera
Grado de Ingeniería Electrónica, Robótica y Mecatrónica

Construcción de un robot (Makeblock), puesta en marcha y control del mismo

Autor: Francisco José Díaz Zamorano
Tutor: Ignacio Alvarado Aldea



Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera
Grado de Ingeniería Electrónica, Robótica y Mecatrónica

Programación de Robot Makeblock con Arduino

Autor:
Francisco José Díaz Zamorano

Tutor:
Ignacio Alvarado Aldea
Profesor titular

Dep. Ingeniería de Sistema y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Proyecto Fin de Carrera: Programación de Robot Makeblock con Arduino

Autor: Francisco José Díaz Zamorano

Tutor: Ignacio Alvarado Aldea

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi familia y amigos

A mis profesores

Agradecimientos

Después de estos intensos años mi época universitaria llega a su fin y quiero agradecer en este trabajo a todos los que han influido para que yo haya podido llegar hasta aquí, tanto en el ámbito personal como en el ámbito lectivo.

En primer lugar, agradecer a toda mi familia por confiar en mí y estar convencidos, en algunas ocasiones hasta más que yo, de que podría con esta etapa de mi vida y su apoyo incondicional, en especial a mis primas Carmen y Cristina por aguantarme y cuidarme tantos años.

Por otro lado, quiero darle las gracias amigos de facultad Edu, Jesupi, Lima, Jesús, Ángel, Jhon y Carreto por toda la paciencia que han tenido conmigo, ya que son los que de verdad han sufrido los malos tragos de estos años, pero sobre todo agradecerles los buenos momentos que me han dado que son los que me de verdad me llevo. Ellos han sido mi familia en Sevilla.

Francisco José Díaz Zamrorano

Sevilla, 2017

Resumen

El trabajo consiste en el control de manera remota de un Brazo/Oruga robótico mediante la comunicación de tres Arduino´s a través del puerto serie (Bluetooth).

El robot pertenece a la empresa MAKEBLOCK y trae de serie un Arduino UNO modificado, lo cual tiene muchas ventajas, como alimentación en todos sus pines o conectores “Plug´n Play”, a cambio de sacrificar el número de pines disponibles. Para solucionar esta falta de pines se ha optado por acoplar al micro del brazo un Arduino GEMMA. Más adelante se verá de manera más detallada este apartado.

Por último se ha acoplado a un arduino UNO (normal), una KiwiBot Basic Shield perteneciente a la asociación sevillana de programación KIWIBOT. Este conjunto es el encargado de hacer de “control remoto” para controlar la oruga y la grúa.

Introducción

La finalidad de este trabajo fue la de abarcar un proyecto desde cero donde entre en juego lo aprendido a lo largo de la carrera a nivel de software y hardware. El proyecto se ha planteado de manera que cualquier persona pueda obtener los mismos resultados siguiendo paso por paso este documento y sin necesidad de ser un experto en la programación de microcontroladores.

Además puede servir como práctica en asignaturas sobre control y electrónica, ya que se plantean problema reales a la hora de implementar software y hardware como la presencia de ruidos y glitches, la importancia del reparto del peso en estructuras o la realización de una PCB.

Índice

Agradecimientos ix

Resumen xi

Introducción xii

Índice xiii

Índice de Figura xiv

Notación xvii

1 Montaje 1

2 Comunicación Serie Bluetooth 11

3 Control Automático PI 16

4 Arduino Makeblock 19

5 Arduino Gemma 23

6 KiwiBot Basic Shield 26

7 Arduino Uno 28

Anexo A: Señales PWM 36

Anexo B: Añadir librerías al IDE Arduino 37

Anexo C:Realización de PCB para potenciómetro y Bluetooth 38

Anexo D:Finales de Carrera 43

Anexo E: Serial Plotter en IDE Arduino 45

Anexo F: Códigos de programación 46

Anexo G: Montaje del robot 54

Referencias 62

Índice de Figura

Ilustración 1. Tipos de montaje 1
Ilustración 2. Piezas necesarias para nuestro montaje 2
Ilustración 3. Modelo real perspectiva 2
Ilustración 4. Modelo real atrás 3
Ilustración 5. Modelo real lateral izquierdo 3
Ilustración 6. Modelo real lateral derecho 4
Ilustración 7. Modelo real frente 4
Ilustración 8. Modelo real abajo 5
Ilustración 9. Modificación en la rueda de transmisión 6
Ilustración 10. Posición del stick de bluetooth 6
Ilustración 11. Distribución de componentes (1) 7
Ilustración 12. Distribución de componentes (2) 7
Ilustración 13. Finales de carrera (1) 8
Ilustración 14. Finales de Carrera (2) 8
Ilustración 15. Servo perspectiva 9
Ilustración 16. Servo arriba 9
Ilustración 17. Servo atrás 10
Ilustración 18. Potenciómetro solidario al eje de giro 10
Ilustración 19. Módulo Bluetooth HC05 11
Ilustración 20. Conexionado Arduino/HC05 12
Ilustración 21. Código para comandos AT Bluetooth 12
Ilustración 22. Fórmula PI 16
Ilustración 23. Esquema control 17
Ilustración 24. Respuesta ante escalón 17
Ilustración 25. Código para control PI 18
Ilustración 26. Esquemático Arduino MeOrion 19
Ilustración 27. Función "Recibir" 21
Ilustración 28. Función "Interpretar" 21
Ilustración 29. Esquemático Arduino Gemma 23
Ilustración 30. Código para Arduino Gemma 25
Ilustración 31. KiwiBot Shield 26
Ilustración 32. Pantalla Led de KiwiBot 27

Ilustración 33. Arduino UNO 28
Ilustración 34. Esquemático Arduino UNO 29
Ilustración 35.Arduino 1 con portapilas 30
Ilustración 36. Arduino 1 con Kiwi 30
Ilustración 37. Arduino con portapilas trasera 31
Ilustración 38. Código para valores del Acelerómetro 32
Ilustración 39. Código para valores del Joystick 33
Ilustración 40. Ejemplo de encapsulado para envío de datos por puerto serie 34
Ilustración 41. Función de gestión de la matriz 35
Ilustración 42. Funcionamiento de PWM 36
Ilustración 43.Esquemáticos del circuito para el módulo Bluetooth 38
Ilustración 44. PCB para módulo Bluetooth 1 40
Ilustración 45. PCB para módulo Bluetooth 2 40
Ilustración 46. PCB para módulo Bluetooth 3 41
Ilustración 47. Esquemático circuito para potenciómetro 41
Ilustración 48. Placa de prototipos para potenciómetro 1 42
Ilustración 49. Placa de prototipos para potenciómetro 2 42
Ilustración 50. Esquemático finales de carrera 43
Ilustración 51. Esquemático Pull-Up/Down 44
Ilustración 52. Código para uso de Serial Plotter 45
Ilustración 53. Montaje 1 54
Ilustración 54. Montaje 2 54
Ilustración 55. Montaje 3 55
Ilustración 56. Montaje 4 55
Ilustración 57. Montaje 5 56
Ilustración 58. Montaje 6 56
Ilustración 59. Montaje 7 57
Ilustración 60. Montaje 8 57
Ilustración 61. Montaje 9 58
Ilustración 62. Montaje 10 58
Ilustración 63. Motaje 11 59
Ilustración 64. Montaje 12 59
Ilustración 65. Montaje 13 60
Ilustración 66. Montaje 14 60
Ilustración 67. Montaje 15 61
Ilustración 68. Montaje 16 61

Notación

=	Igual
<=	Menor o igual
>=	Mayor o igual
RGB	Red, Green, Blue
PCB	Tarjeta de cirucitos impresa
Tx	Transmisión
Rx	Recepción
PC	Ordenador
IDE	Entorno de desarrollo integrado
Comandos AT	Comandos “Atención”
NL	Nueva línea
CR	Retorno de carro
PI	Proporcional e integral
Motor DC	Motor de corriente continua
I2C	Inter-Integrated Circuit
PWM	Power Width Modulation
USB	Universal Serie Bus
JST	Japan Solderless Terminal

1 MONTAJE

El kit seleccionado tiene muchas configuraciones disponibles, se eligió la grúa por ser el más completo de todos en el uso de actuadores y sensores.

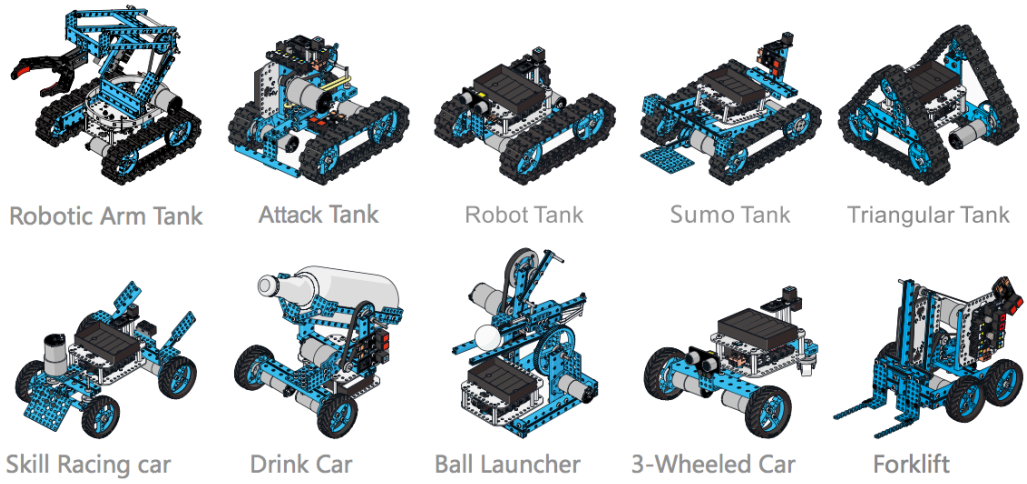


Ilustración 1. Tipos de montaje

En primer lugar se montó el robot tal y como viene en el manual de instrucciones y luego se procedió a realizar una serie de modificaciones para que las funciones de este sean las deseadas.

En el kit se incluyen muchas piezas y dispositivos, de los cuales solo se usan una parte:

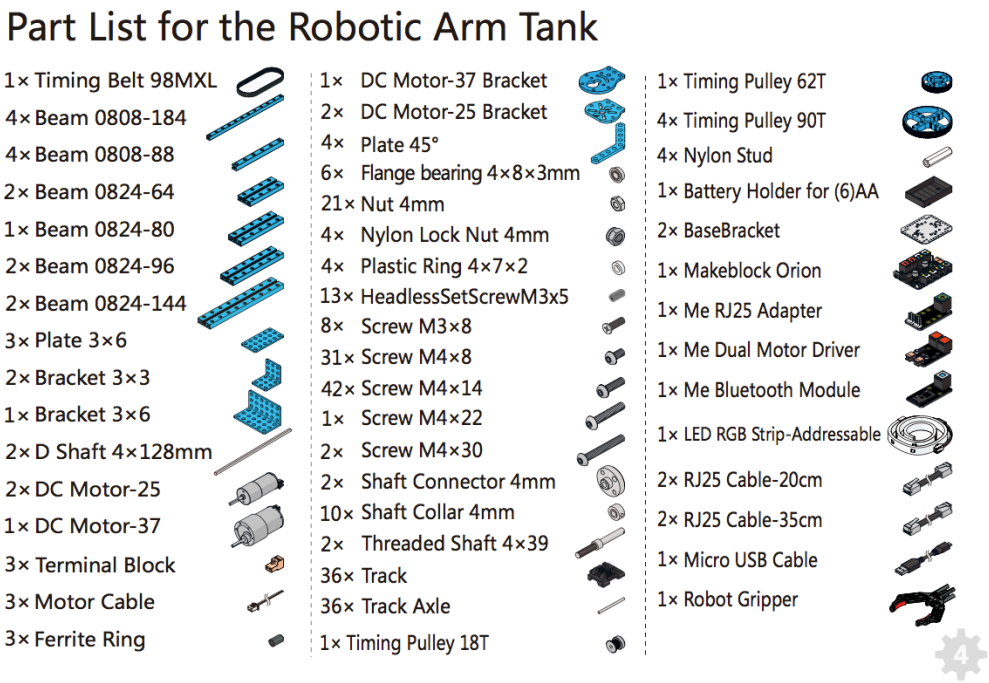


Ilustración 2. Piezas necesarias para nuestro montaje

El montaje general del robot se ha extraído del manual de instrucciones del kit, ya que viene muy bien explicado. Para una mejor claridad en la memoria, las ilustraciones con el montaje del robot se han adjuntado en el anexo F al final de la memoria.

El modelo real tiene una serie de diferencias respecto al que puede verse en la imagen superior.

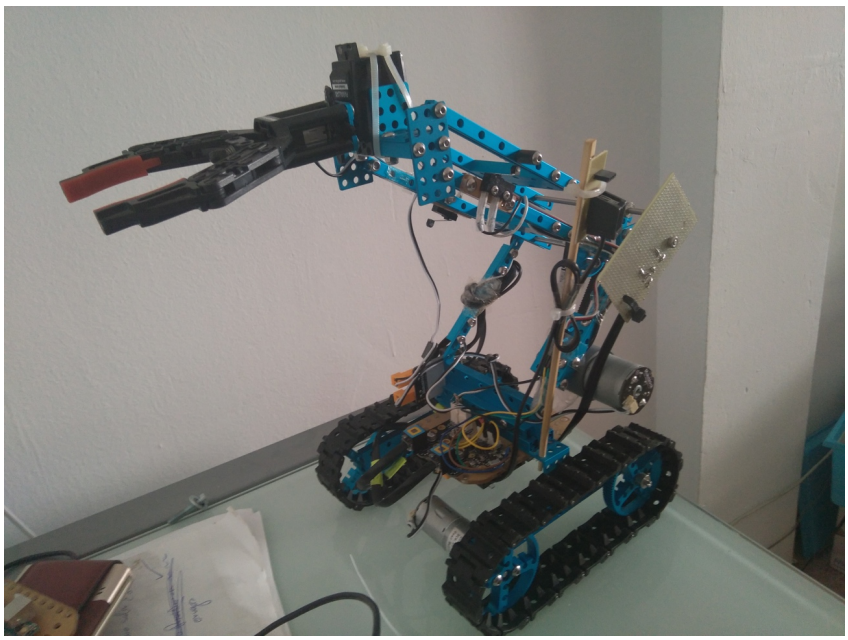


Ilustración 3. Modelo real perspectiva

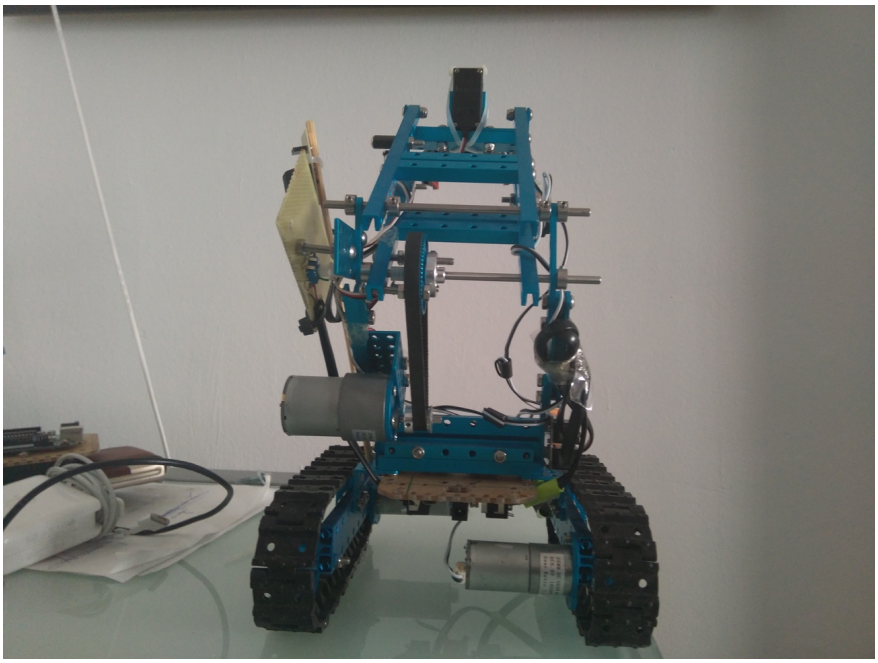


Ilustración 4. Modelo real atrás

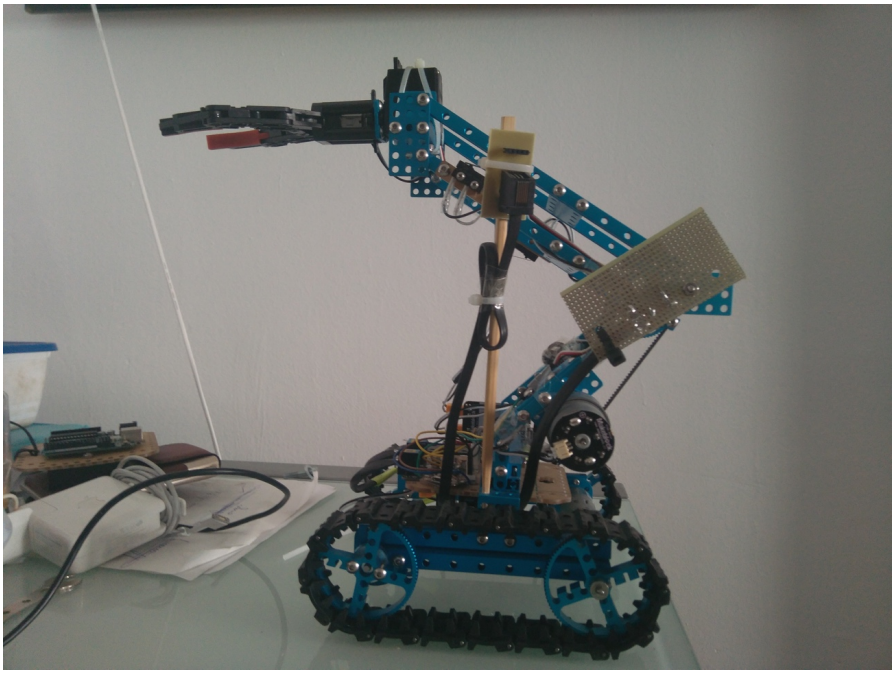


Ilustración 5. Modelo real lateral izquierdo

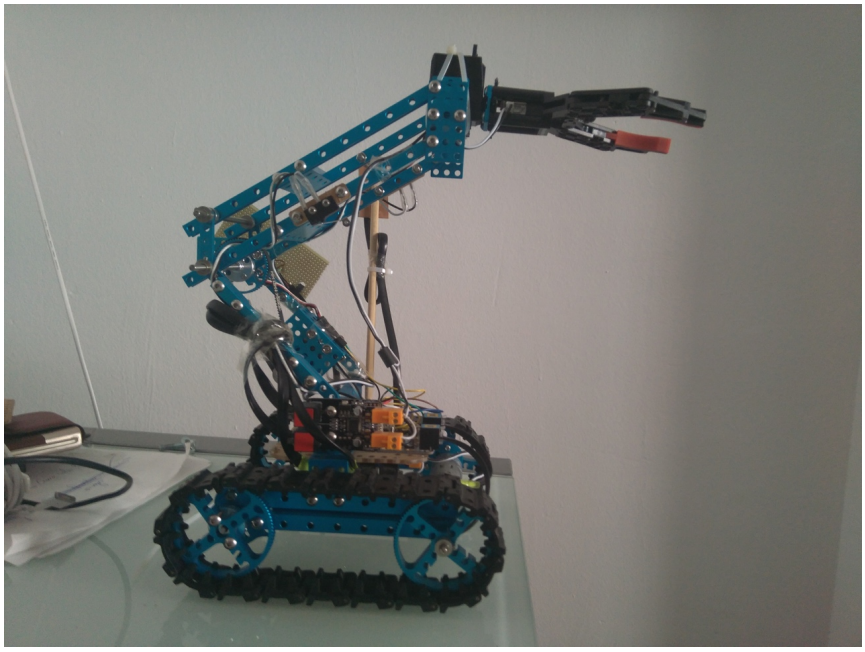


Ilustración 6. Modelo real lateral derecho

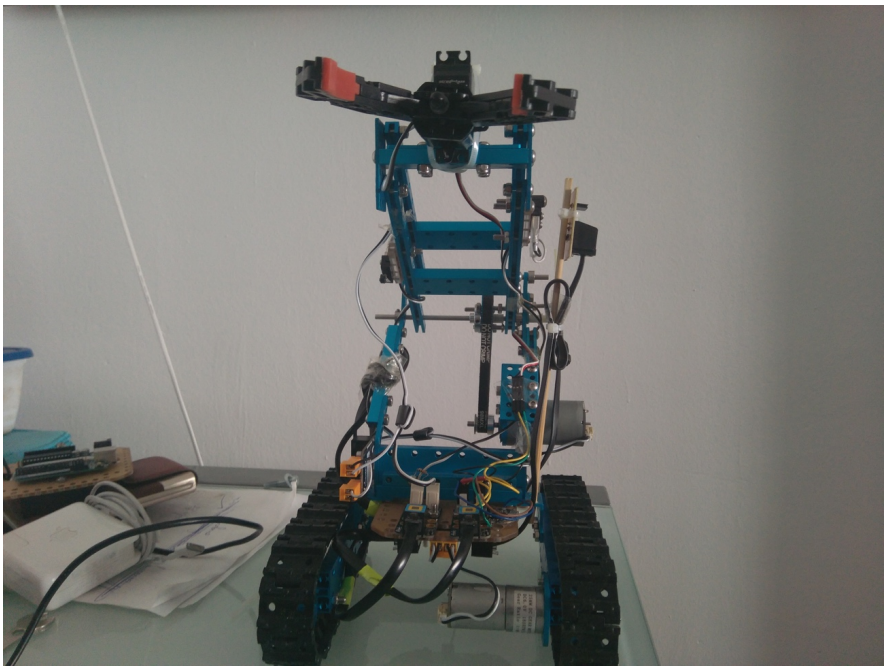


Ilustración 7. Modelo real frente

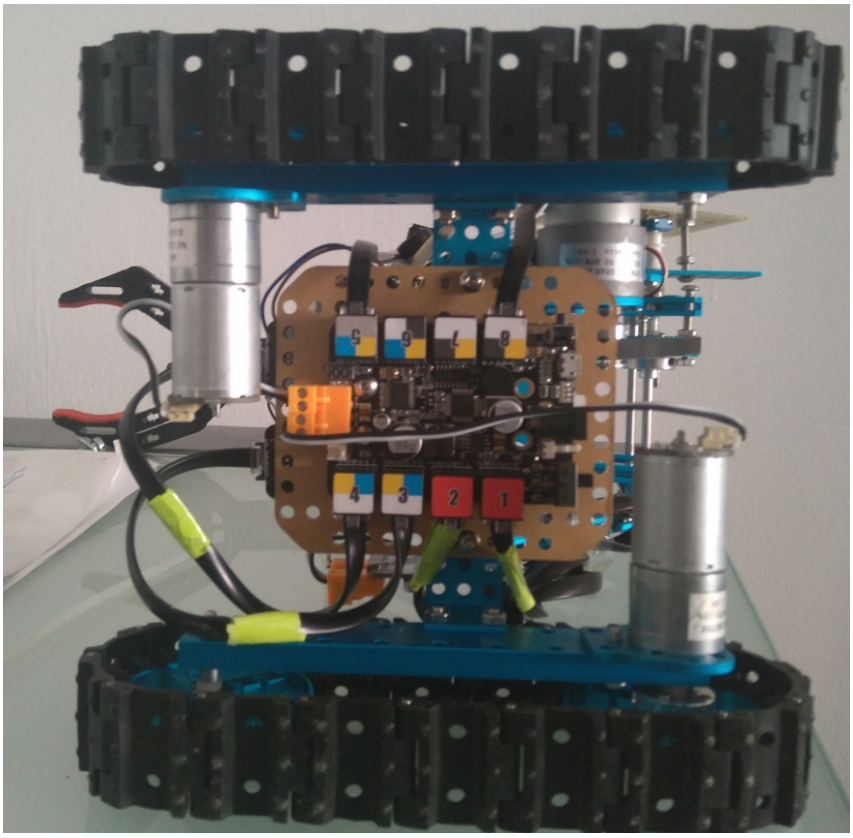


Ilustración 8. Modelo real abajo

Haciendo hincapié en las diferencias:

- Se ha optado por prescindir de la tira de led RGB, ya que es algo meramente decorativo y era necesario tener el máximo de pines disponibles.

- Se ha añadido un tornillo más a la rueda dentada del eje para una mejor sujeción de esta.

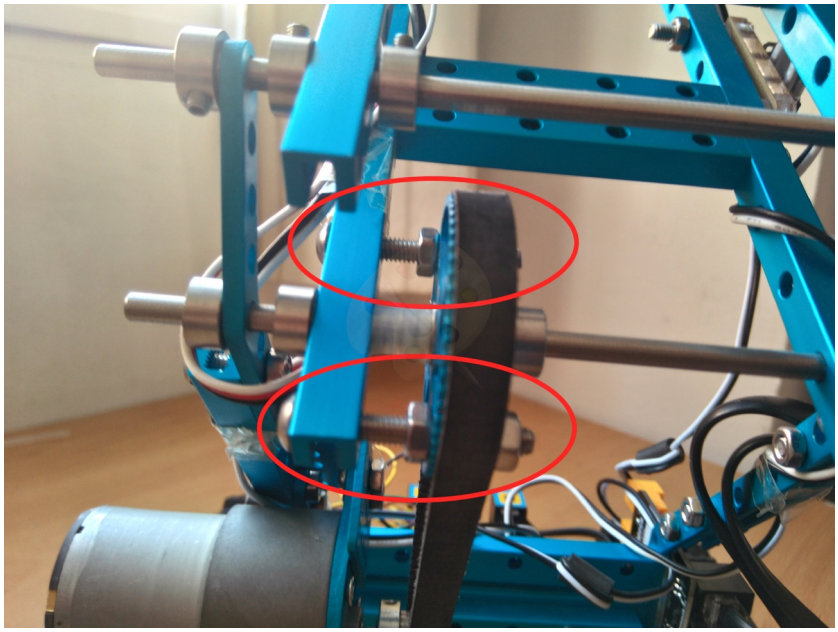


Ilustración 9. Modificación en la rueda de transmisión

- El módulo bluetooth va sujeto a un stick de madera a modo de antena.

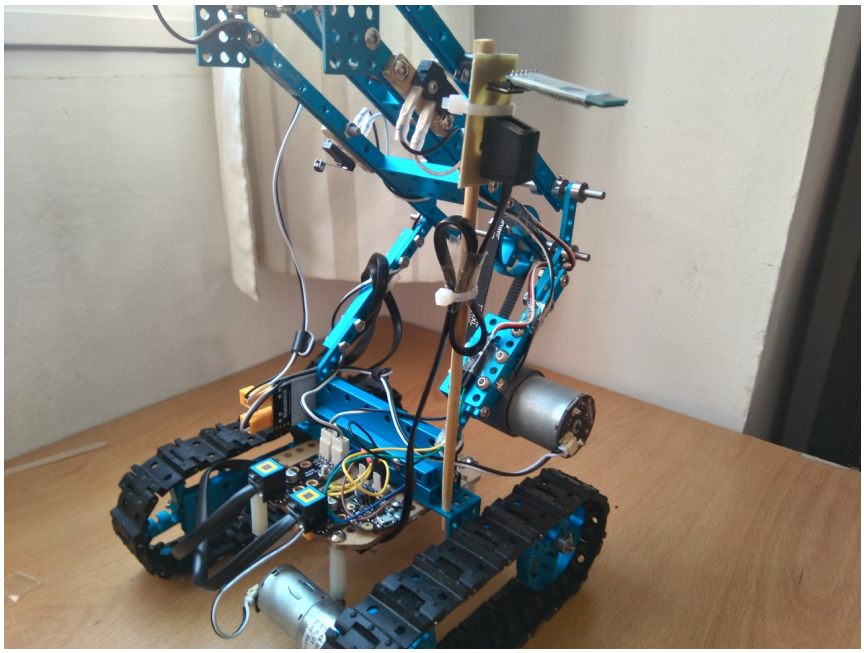


Ilustración 10. Posición del stick de bluetooth

- La distribución de los dispositivos electrónicos es distinta, se ha acoplado todo en la base del robot para intentar distribuir todo el peso en la parte inferior y darle más estabilidad.

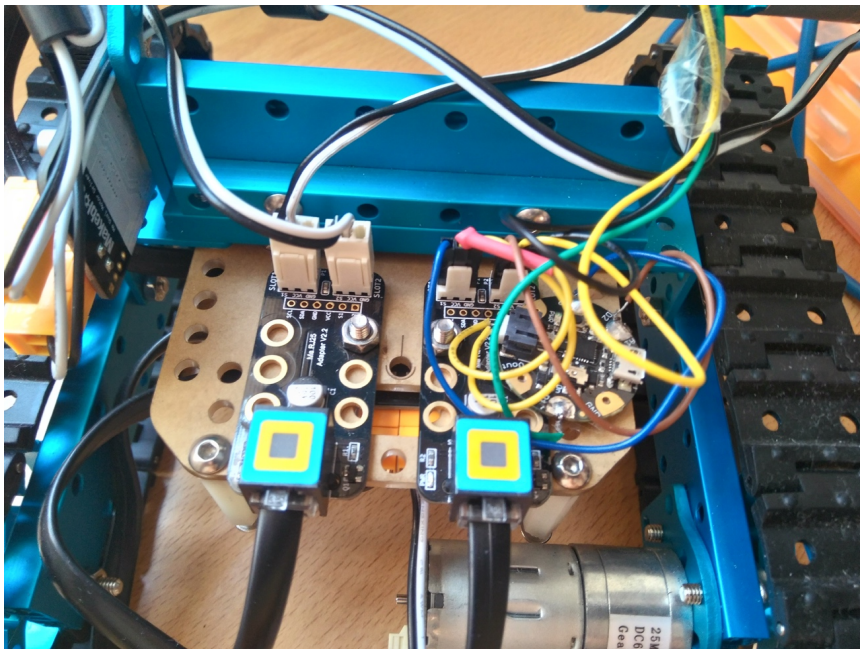


Ilustración 11. Distribución de componentes (1)

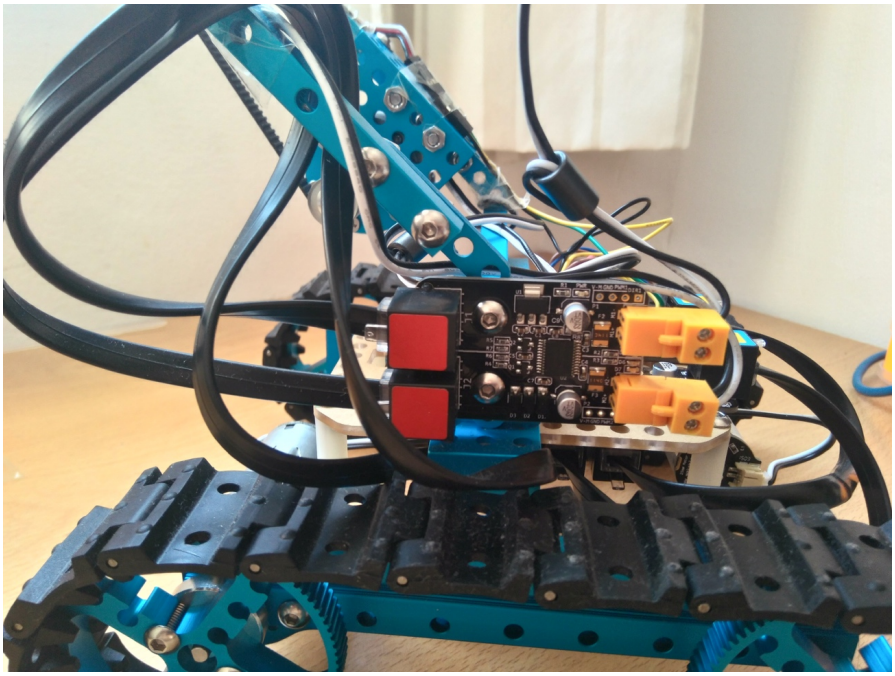


Ilustración 12. Distribución de componentes (2)

- Se han añadido los finales de carrera en la parte superior de la grúa.

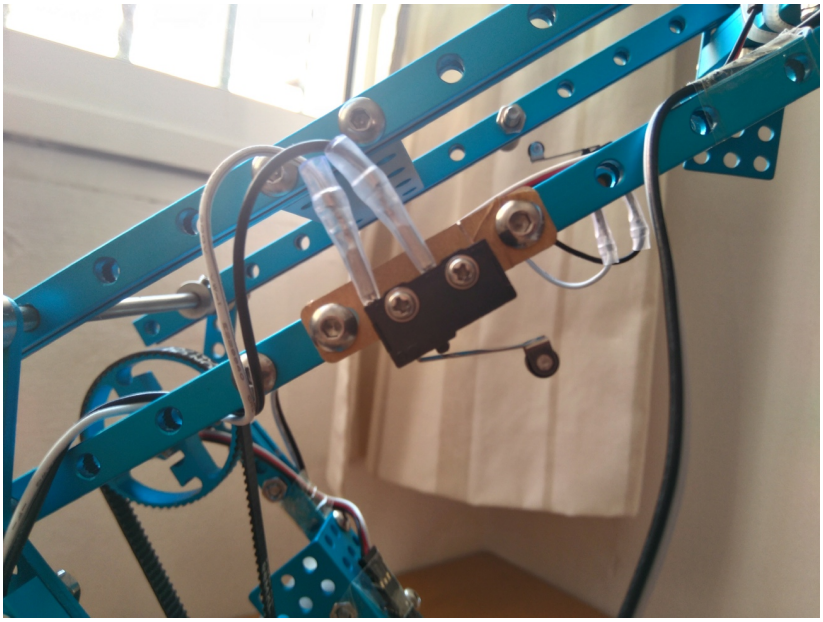


Ilustración 13. Finales de carrera (1)

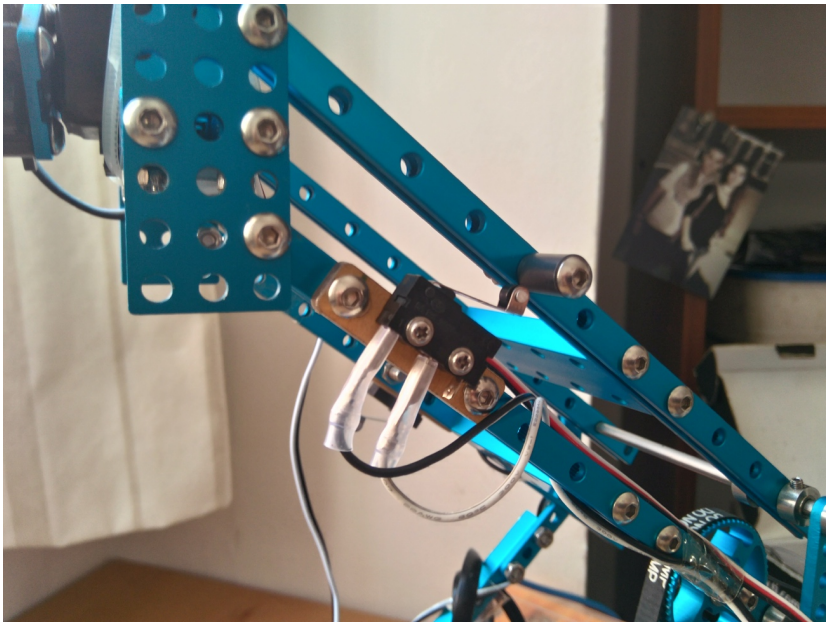


Ilustración 14. Finales de Carrera (2)

- En el extremo de la grúa se ha acoplado un servo que hace posible que la garra gire.

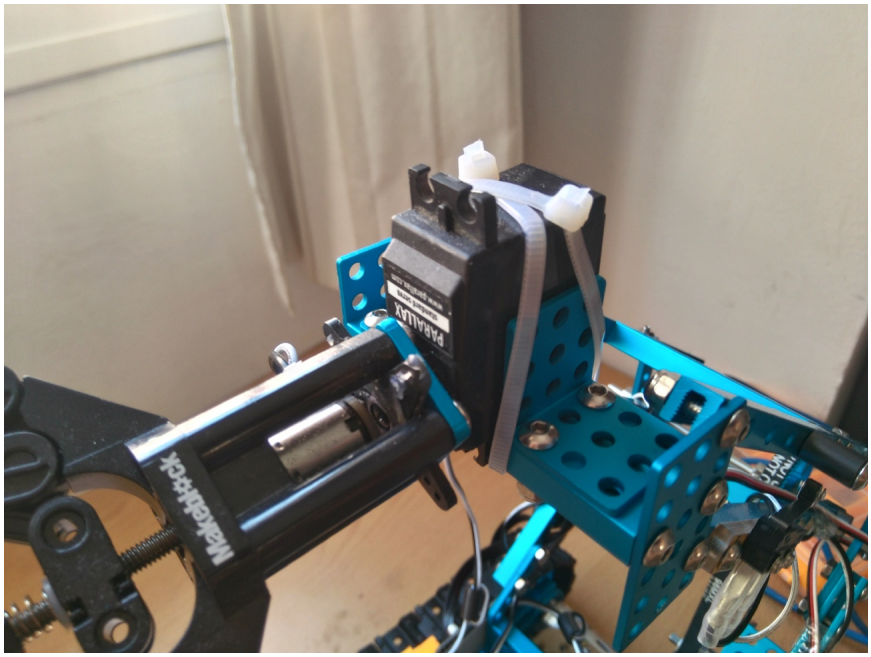


Ilustración 15. Servo perspectiva

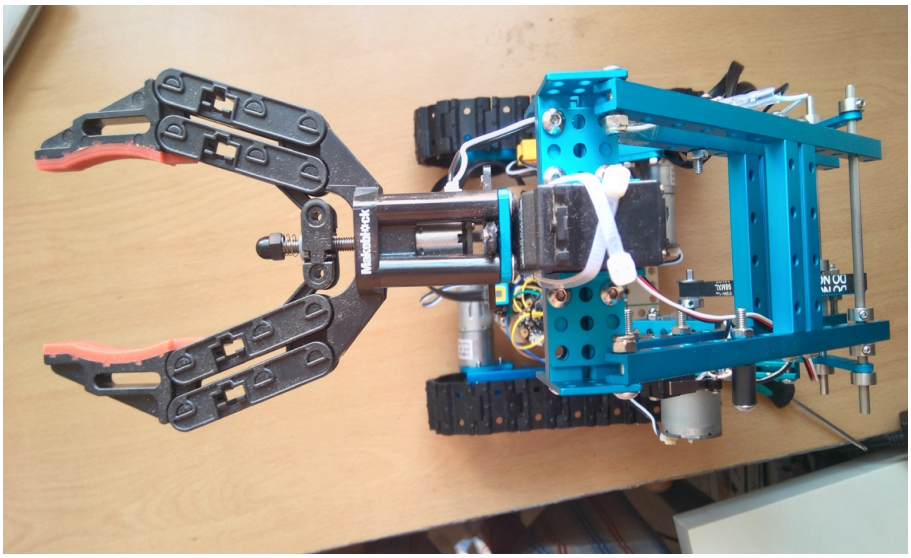


Ilustración 16. Servo arriba

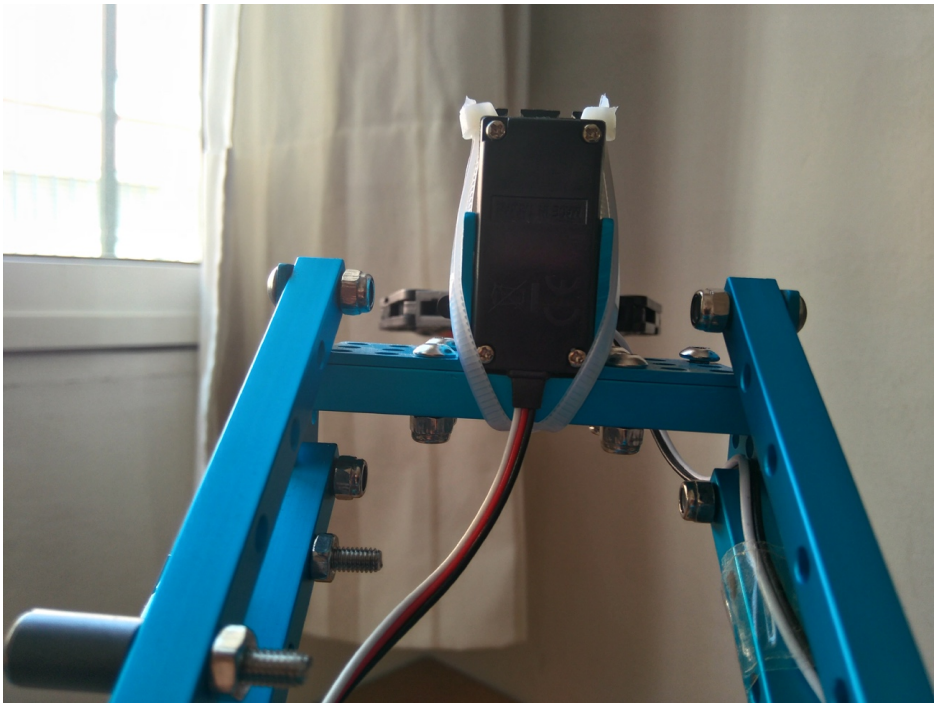


Ilustración 17. Servo atrás

- Se ha acoplado una placa de pruebas solidario al eje de giro de la grúa con un potenciómetro para conocer la posición en la que se encuentra.

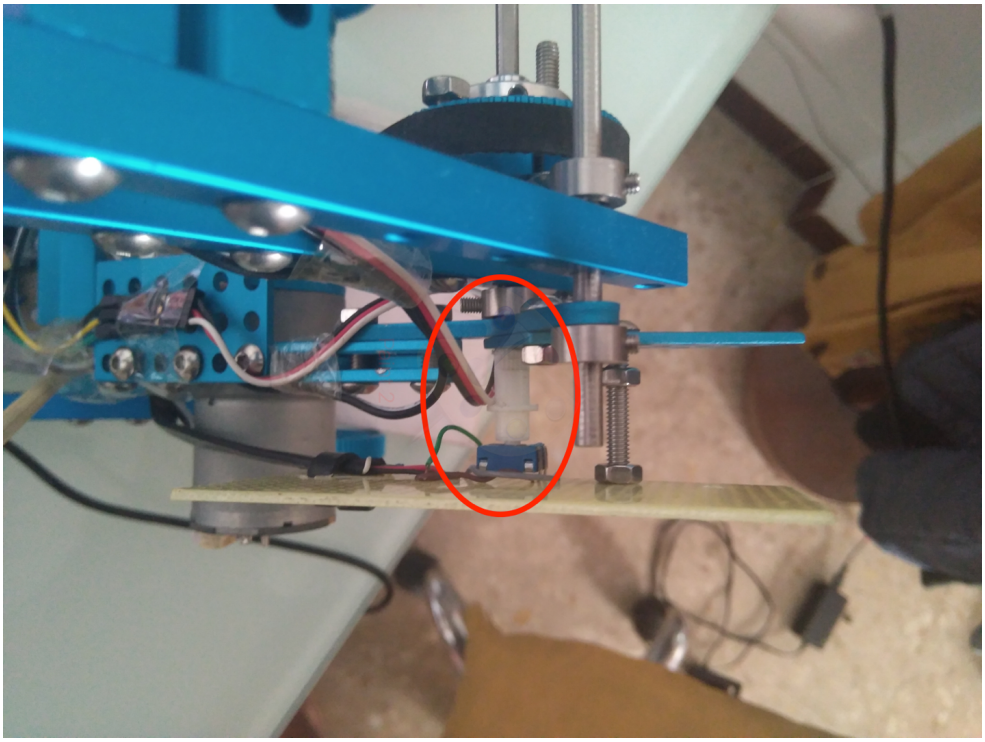


Ilustración 18. Potenciómetro solidario al eje de giro

2 COMUNICACIÓN SERIE BLUETOOTH

Para la conexión entre las placas se han usado dos módulos Bluetooth HC05, uno configurado como Maestro conectado a la placa KIWI y el otro, como esclavo, conectado al robot, se ha optado por esta opción ya que se trata de dispositivos económicos que vienen preparados para ser cableados o conectados a un microcontrolador de manera sencilla.

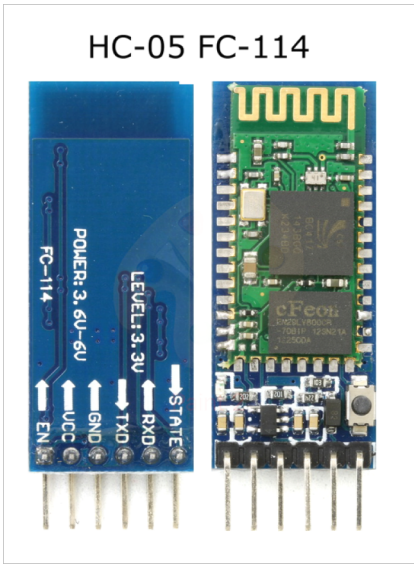


Ilustración 19. Módulo Bluetooth HC05

En primer lugar hay que configurar el Bluetooth HC-05 como maestro ya que de fábrica viene configurado como esclavo. El HC-05 tiene un modo de comandos AT que puede activarse de varias maneras. La primera, presente en los modelos más antiguos, es el uso del pin llamado KEY, que debe ponerse a nivel alto mientras se enciende el dispositivo. En los modelos nuevos el módulo dispone de un botón que debe ser presionado mientras encendemos el dispositivo. Mediante ambos métodos sabremos que estamos en modo de recepción de comandos porque el led del módulo parpadeará con una frecuencia de 1 segundo.

Es muy importante el correcto conexionado del módulo con el micro, el pin RX del módulo tiene que ir al pin TX de Arduino y viceversa, es decir, que se crucen como se muestra en la siguiente imagen:

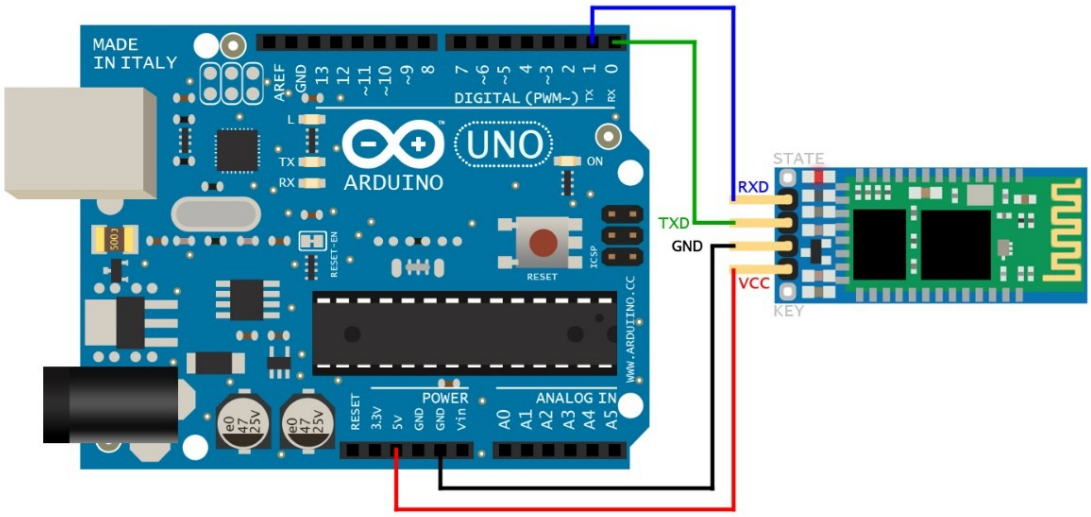


Ilustración 20. Conexionado Arduino/HC05

Una vez entremos en este modo, es necesario cargar un programa que lee datos enviados por el puerto serie del PC a través del IDE de arduino y se lo envía hacia los pines RX y TX del módulo HC-05. El código usado es el siguiente:

```
//Librería para definir los pines RX y TX libremente , si no se
//definiera también funcionaría se forzarían a RX=0 y TX=1
#include <SoftwareSerial.h>
SoftwareSerial BT1(0, 1); // RX | TX

void setup()
   {
     Serial.begin(9600);
     Serial.println("Arrancando modulo HC-05");
     delay (500) ;
     Serial.println("Esperando comandos AT:");
     BT1.begin(38400);

   }

void loop(){
//Si llega un dato procedente del BT enviar al terminal serie
     if (BT1.available())
            Serial.write(BT1.read());
//Si se manda un dato por el terminal enviar al BT
      if (Serial.available())
            BT1.write(Serial.read());
   }
```

Ilustración 21. Código para comandos AT Bluetooth

A continuación abrimos el terminal de Arduino (o cualquier otro), es importante marcar la opción en la esquina inferior izquierda de "Ambos NL & CR", esto quiere decir que cada vez que se mande un mensaje por el puerto serie se iniciará una nueva línea y un retorno de carro, y la velocidad que hayamos elegido para la comunicación, 38400 baudios en nuestro caso.

Una vez configurado el terminal, se puede empezar a configurar el Bluetooth. Lo primero que se debe hacer es comprobar que el módulo está realmente en el modo de comandos AT, para ello escribir en el terminal:

- Enviar: AT
- Repuesta: OK

En caso de que no se obtuviera la respuesta de "OK" habría que verificar los pasos anteriores y el conexionado.

En el caso de que recibamos la respuesta de "OK", podremos continuar mandando comandos para cambiar la configuración del módulo. A continuación se ofrece una lista con los parámetros que se pueden modificar:

- Nombre del módulo. Por defecto se llama "HC-05", para cambiarlo hay que mandar el comando:

  Enviar: AT+NAME=<Nombre>

  Recibiendo "OK" como respuesta.

- Cambiar código de vinculación. Por defecto el código de vinculación es "1234", para cambiarlo:

  Enviar: AT+PSWD=<Pin>

  Se recibe "OK" si ha habido éxito en la operación, para saber cuál es el código actual: AT+PSWD?

- Configurar la velocidad de comunicación. La velocidad por defecto es 9600 baudios, con 1 bit de parada y sin paridad, para cambiar estos parámetros:

  Enviar: AT+UART=<Baud>,<StopBit>,<Parity>, donde:

  - Baud: Velocidad de la comunicación (4800,9600,19200,…).
  - StopBit: Bit de parada, 0 para 1 bit y 1 para dos bits.
  - Parity: Bit de paridad, 0 sin paridad, 1 para paridad impar y 2 para paridad impar.

Para saber la configuración actual: AT+UART?

- Configurar Rol. Por defecto el módulo viene configurado como esclavo, para cambiarlo:

  Enviar: AT+ROLE=<Role>, donde Role puede ser 0 para esclavo y 1 para maestro.

  Para saber el modo actual: AT+ROLE?

- Configurar el modo de conexión. Esta configuración se aplica solo cuando el bluetooth esta funcionando como maestro, se utiliza para "decirle" si se va a conectar a un dispositivo en concreto o a cualquiera que esté disponible.

  Enviar: AT+CMODE=<Mode>, donde Mode:

  - 0: Conectarse a un dispositivo con una dirección específica.
  - 1: Conectarse a cualquier dispositivo disponible.

  Para saber la configuración actual: AT+CMODE?

- Especificar dirección de dispositivo a conectar. Para poder decirle a que dispositivo esclavo se va a conectar nuestro módulo tiene que estar configurado como maestro y CMODE=0.

  Enviar: AT+BIND=<Dirección>, donde Adress es la dirección del dispositivo al que nos vamos a conectar con la siguiente forma: 1234,56,ABCDEF.

  Para saber la configuración actual: AT+BIND?

- Obtener la dirección de nuestro módulo Bluetooth.

  Enviar: AT+ADDR?, donde se recibirá de la forma 1234,56,ABCDEF.

Del otro módulo HC05 solo habría que obtener la dirección para poder introducirla en el master, esto se hace para que ambos módulos se emparejen directamente y el maestro no se conecte a otro dispositivo esclavo cercano de manera indeseada. Los módulos se deben emparejar de manera automática una vez se alimenten, cuando se hayan emparejado las luces led de ambos módulos parpadearan a menor frecuencia.

Una vez realizado todo esto y con nuestros módulos configurados como es debido, podemos pasar a realizar la comunicación entre las dos placas, esta conexión consiste en mandar comandos, una vez los módulos se hayan emparejado, a través del puerto serie de las placas como si estuvieran conectadas directamente por cable.

Para realizar la comunicación en primer lugar hay que inicializar el puerto serie, esto se hace declarando dentro de la función de configuración lo siguiente:

```
void setup(){
    Serial.begin(9600);}
```

Luego dentro de la función donde va todo el código (void loop), se usan las siguientes funciones:

- Serial.print (num): Manda por el puerto serie el valor de la variable num.
- Serial.println (num): Manda el valor de num e inserta una línea nueva.
- Serial.print ('\t'): Manda una tabulación.
- Serial.available (): Hay datos recibiéndose.
- Serial.read(): Asigna a una variable el valor que haya en el puerto serie.

Para establecer la conexión del puerto serie es necesario asignar como Rx y Tx a dos pines, en nuestro caso hemos usado los pines 0 y 1 respectivamente.

3 CONTROL AUTOMÁTICO PI

La posición de la grúa se consigue a través del control del ángulo de giro, por medio de un PI (Proporcional e integral) programado en una función de Arduino. En nuestro caso con un control P hubiera sido suficiente ya que nos asegura un error en régimen permanente nulo, pero nuestro sistema tiene una perturbación constante , que es la carga del brazo, se añade un término integral.

La función de transferencia en velocidad del sistema es:

$$G(s) = \frac{K}{1 + \tau s}$$

Para calcular la función en posición integramos la anterior:

$$G(s) = \frac{K}{s(1 + \tau s)}$$

Sin entrar demasiado en teoría, tenemos que la ecuación de control es:

$$u(t) = \underbrace{K_p e(t)}_{\text{Proportional}} + \underbrace{K_i \int_0^t e(\tau)d\tau}_{\text{Integral}}$$

Ilustración 22. Fórmula PI

Donde:

- e(t): Error de la señal.
- u(t): Salida del controlador.
- Kp: Ganancia proporcional
- Ki: Ganancia integral.

Muy por encima, el funcionamiento de estos bloques consiste en :

- Proporcional: Aplica mayor señal de control cuanto mayor es el error.
- Integral: Disminuye y elimina el error en estado estacionario. Este bloque actúa cuando la referencia y el valor de la variable son distintos, integrando la desviación en el tiempo y añadiéndola a la acción proporcional.

Se ha decidido no añadir bloque derivativo debido a que los ruidos que afectan a la medida del ángulo son amplificados por este término. Un esquema de lo que se ha realizado en el brazo:

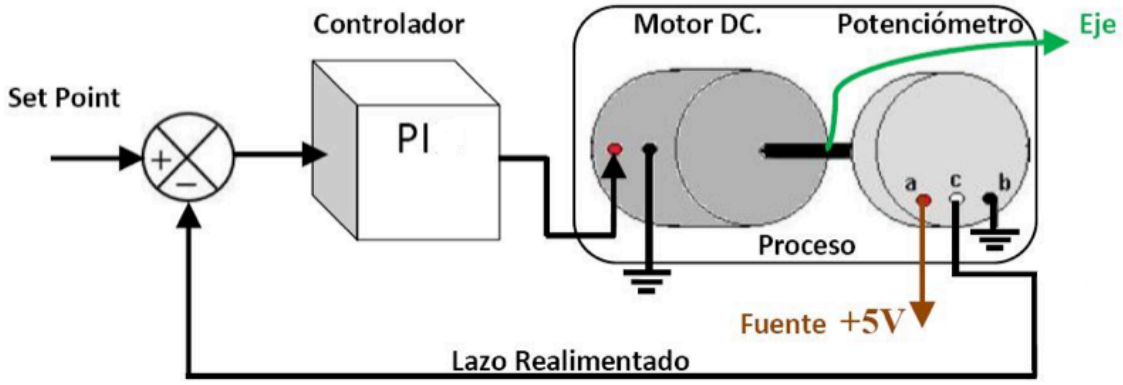


Ilustración 23. Esquema control

Donde:

- SetPoint (Referencia): Dato pasado por el Kiwi
- Controlador PI: Su salida controlar el motor
- Motor DC (Sistema a controlar): Motor que mueve la grúa.
- Potenciómetro (Realimentación): Potenciómetro acoplado al eje de la grúa.

Los parámetros de nuestro controlador se han sacado mediante prueba y error ya que se trata de un sistema sencillo. Para probar el control se le ha dado un cambio en la referencia en escalón y se ha obtenido lo siguiente:

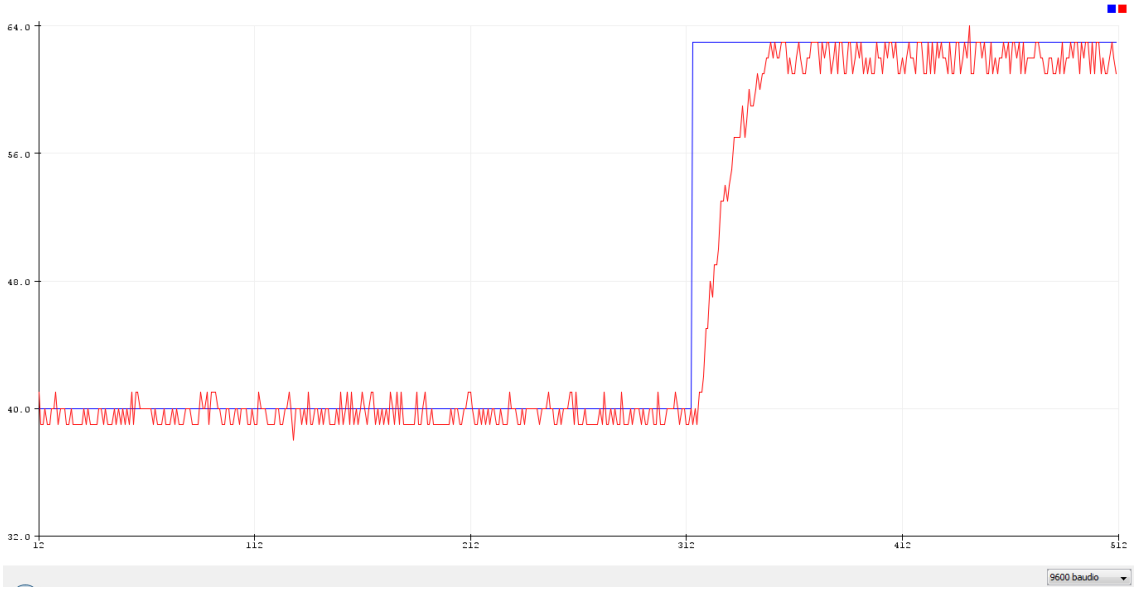


Ilustración 24. Respuesta ante escalón

Se observa que la posición se queda un poco por debajo de la referencia, esto se debe a la zona muerta, ya que el motor para muy bajas señal de control no se mueve, por lo que no llega.

El código usado para la función es el siguiente:

```
#include <Wire.h>//Librería para protocolo I2C
#include <MeOrion.h>//Librería con todas las funciones de los componentes de Makeblock

MeDCMotor motor_1(1);//Motor de grua
int pot;
int e,sum_e=0; //Señal de error y sumatorio
int sal;//Salida de actuación
float Kp=3.5;
float Ki=2;
float Tm=0.1;
float integral;// Termino integral
int ref;
int Ejey=400;//QUITAR PUESTO QUE YA ESTA DEFINIDA EN EL PROGRAMA REAL
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  pot=analogRead(A7);
  pot=map(pot,234,604,0,90);//Mapeamos los valores del  potenciometro.
  ref=map(Ejey,260,460,0,90);//Mapeamos los valores del Eje y del acelerómetro
  control ();
  motor_1.run(sal);
  Serial.print(ref);
  Serial.print(",");
  Serial.println(pot);
  delay(50);
}

void control (){
  //Calculo de la señal de control:
  e=ref-pot;//Caclulo del error
  sum_e=e+sum_e;//Actualizamos el sumatorio del error
  integral=Ki*sum_e*Tm; //Calculamos termino integral
//Acotamos el término integral para controlar el WindUp
  if(integral>=40){
    integral=40;
  }
  if(integral<=-10){
    integral=-10;
  }
  sal=Kp*e+integral;
}
```

Ilustración 25. Código para control PI

4 ARDUINO MAKEBLOCK

El Arduino usado para el control de las orugas/brazo se trata de un Arduino Uno, llamado MeOrion, modificado por la empresa Makeblock, que es la que también facilita el robot. La placa base Orion posee dos drivers para motores DC, un buzzer y 8 conectores RJ25 de 6 hilos, para realizar las conexiones con los dispositivos que vienen con el pack. Esto es una ventaja ya que permite una rápida conexión de todos los módulos, pero se pierden muchos pines del Arduino Uno original, como puede verse en la siguiente figura:

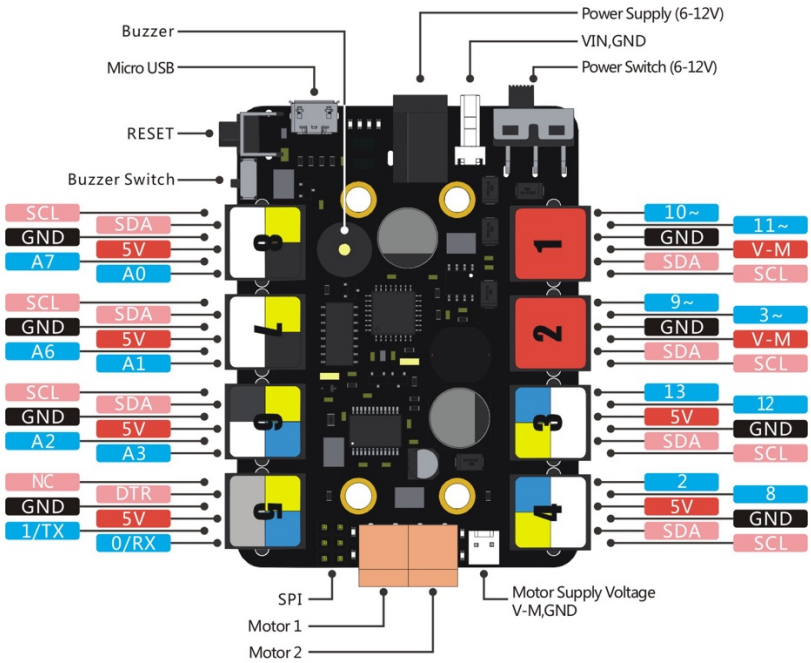


Ilustración 26. Esquemático Arduino MeOrion

Como puede apreciarse en la imagen, los puertos están marcados con un código de colores, este código de colores indica las características de cada pin, siendo:

- Rojo: Salida de 6 a 12 voltios, se usa para colocar los motores DC de las orugas.
- Amarillo: Puerto de de entrada/salida digital simple.
- Azul: Puerto de entrada/salida digital doble.
- Gris: Puerto usados en la comunicación serie, es decir Rx y Tx.
- Negro: Puerto analógico.
- Blanco: Puerto para comunicación I2C.

En nuestro poyecto el reparto de pines se ha realizado de la siguiente manera:

- Puerto 1: Motor Grúa
- Puerto 2: Motor Garra
- Puerto 3: Interrutores finales de Carrera
- Puerto 4: Gestión con Gemma para Servo
- Puerto 5: Módulo Bluetooth HC05
- Puerto 8: Potenciómetro para control

Los motores de las orugas están conectados a los puertos para dicho fin llamados Motor 1 y Motor 2.

La placa se programa mediante el puerto microUSB y un PC, puede usarse el IDE de Arduino, aunque también puede programarse mediante el lenguaje Scratch, que es un lenguaje visual, usado para acercar a los más jóvenes y recién iniciados en la programación de microcontroladores. Además, la empresa Makeblock ha creado su propio compilador (mBlock) donde ambas formas de programar se combinan.

MeOrion puede alimentarse a través del puerto microUSB o mediante una batería con una tensión DC de 6 a 12 Voltios a través de el puerto Jack, puede soportar una corriente máxima de 3A. Dispone de protección contra sobretensiones y sobrecorrientes.

Antes de programar por primera vez con el IDE de Arduino la placa Orion, es necesario instalar todos los drivers y las librerías para que el compilador reconozca el microcontrolador, estos documentos y archivos pueden encontrarse fácilmente en el foro oficial de “Makeblock España” o en su web.

Nuestro programa se encarga, en líneas generales, de recibir los datos vía Bluetooth procedentes del Arduino con el KiwiBot acoplado y, dependiendo de los comandos recibidos, realizar las acciones pertinentes. En un primer momento se optó por pasar desde el kiwi solamente comandos, pero para un mejor funcionamiento de las orugas y el control de la grúa se decidió pasar, además, los datos leídos del joystick y del acelerómetro.

Hacer esto no es algo trivial, ya que arduino no sabe qué datos corresponden a qué variables o cuando empieza el “paquete” de datos que le enviamos, por lo que hay que hacer un protocolo para que sepa cuando tiene que empezar a leer datos del buffer y cuando acaba el paquete. Esto se consigue añadiendo un carácter inicial y otro final al paquete. A continuación, se separan nuestros datos por comas para saber a qué variables corresponden los datos, esto se ve de manera más clara en el punto que se explica el Arduino con el Kiwi algo más adelante.

El procesode adquisición de datos se hace a través de dos funciones que se encuentran en el código del Arduino del brazo. Una se encarga de recibir y guardar los datos (Recibir) y otra de interpretarlos (Interpretar). El código está explicado a continuación en los comentarios.

- Función recibir:

```
void Recibir (){
  static boolean recvInProgress=false; //Bandera para saber quee estamos recibiendo datos
  static byte ndx=0;//Indice del vector
  char startMarker='<';//Marca de inicio
  char endMarker='>';//Marca de final de trama
  char rc;//Componente del vector que lee el dato del puerto serie

  while (Serial.available()>0 && newData==false){
    rc=Serial.read();//Dato que se lee del buffer
    if (recvInProgress==true){//Bandera de lectura de datos activada
      if(rc!=endMarker){//Si es igual que ">"
        receivedChars[ndx]=rc;//Guardamos en el vector el dato
        ndx++;//Incrementamos el indice del vector
        if(ndx>=numChars){// Si llenamos el vector antes de que llegue
          ndx=numChars-1;//el caracter de fin, sobreescribimos el ultimo
        }
      }
     else{
      receivedChars[ndx]='\0';//Cerramos la cadena
      recvInProgress=false;//Bajamos la bandera de que estan llegando datos
      ndx=0;//Reiniciamos indice
      newData=true; //indicamos que hay un nuevo dato que interpretar
      }
    }
   else if (rc==startMarker){//Llega el caracter de inicio de lectura
    recvInProgress=true;//Activamos la bandera par que empiece a leer.
   }
  }

}
```

Ilustración 27. Función "Recibir"

- Función Interpretar:

```
void Interpretar(){

  //La funcion strtok permite cortar cadenas en subcadenas mediante separadores, reemplazando estos separadores
  //por caracteres nulos \0.
  //La funcion strcpy sirve para copiar cadenas en variables
  //La funcion atoi sirve para convertir tipos de datos, en este caso de caracteres a enteros

  char*strtokIndx; // Lo usa la funcion strtok como indice

  strtokIndx=strtok(tempChars,",");//Cogemos los datos del vector recibido hasta la primera coma
  strcpy(caracter,strtokIndx);//Copia el dato en la funcion caracter

  strtokIndx=strtok(NULL,",");//Continua desde donde se dejo la ultima vez
  Ejex=atoi(strtokIndx);//convertimos el dato a entero

  strtokIndx=strtok(NULL,",");
  Ejey=atoi(strtokIndx);//convertimos el dato a entero

}
```

Ilustración 28. Función "Interpretar"

El método de transmisión para la subida y bajada de la grúa es una correa dentada desde el motor hasta una rueda dentada situada en el eje de giro de esta, para que la correa no sufra demasiado daño y así extender su vida útil lo máximo posible, se han colocado interruptores final de carrera en los extremos superior e inferior de la grúa para que el motor se pare en el caso de que se activen, estos están explicados en el anexo 1 al final de la memoria.

Al usar todos los PWM´s disponibles con los motores DC se decidió usar un Arduino Gemma (explicado en el siguiente punto) para gestionar a través de dos señales digitales una señal PWM y poder controlar el ángulo de giro del servo acoplado en la base de garra.

5 ARDUINO GEMMA

Cuando se abarcó el problema de la falta de PWM´s del Arduino del robot se estudió la posibilidad de cambiar esta por un Arduino más grande y potente con un PinOut mayor, el problema es la falta de compatibilidad de los componentes que ya habíamos adquirido con otro Arduino, por lo que se optó acoplar un Arduino Gemma, que tiene unas dimensiones pequeñas, al robot y a través de dos pines digitales controlar los PWM´s que posee Gemma. En primer lugar se hará una breve presentación sobre este micontrolador

Arduino Gemma es un microcontrolador basado en ATtiny85. Posee 3 pines digitales de entrada/salida, dos de ellas pueden programarse como Pwm´s y una como entrada analógica. Ofrece una corriente por cada pin de 20 mA.

Su tensión de funcionamiento es 3.3V y puede alimentarse a través del puerto microUSB o con una batería con un conector JST. Además de las E/S, este micro tiene dos pines de alimentación:

- Vout: Se alimenta directamente a la toma de USB/batería sin pasar por el regulador para ofrecer la corriente que la alimentación pueda proporcionar (alrededor de 500 mA).
- 3Vo: Pasa por el regulador de tensión y ofrece hasta 150 mA a una tensión de 3.3 V.
- GND: Arduino Gemma posee un pin de tierra común conectado a la tierra USB y al regulador de potencia.

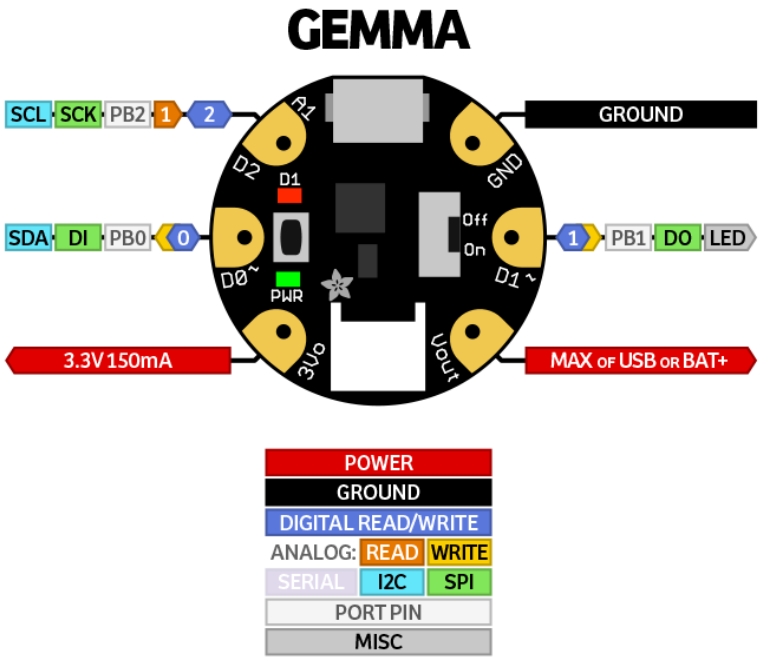


Ilustración 29. Esquemático Arduino Gemma

El método de programación es similar al de un arduino normal pero con algunos pequeños matices. En primer lugar hay que instalar los drivers para que el IDE de Arduino reconozca nuestro dispositivo, esto se

debe a que Gemma pertenece a la empresa Adafruit, si se tiene instalado la última versión del IDE quizás este paso no sea necesario.

A la hora de subir nuestro programa hay que cambiar en el apartado Herramientas/Placa/Arduino Gemma para indicar al IDE que vamos a trabajar con dicha placa. Además este micro no tiene disponible el puerto serie a través del USB por lo que hay cambiar la forma de subir nuestro programa, para ello Herramientas/Programador/USBtinyISP o Arduino Gemma, depende de la versión de IDE instalada en el PC. No es necesario seleccionar ningún puerto.

Nuestro Gemma se dedica a controlar el giro de la garra a través de un servo colocado en la base de la extremidad. Esto se hace a través de dos entradas digitales y una salida PWM. Las dos entradas digitales provienen del Arduino del robot, cada señal representa la dirección en la que queremos que gire la muñeca, es decir, derecha o izquierda, cabe destacar que estas dos señales no pueden estar a nivel alto a la vez, esto se tiene en cuenta en el Arduino del robot. Cada vez que estas señales se activan se incrementa, o decrementa, el valor que pasamos por el PWM al servo, que corresponde con su posición, estos valores tienen un rango de 10 a 250, en realidad va de 0 a 255, pero tomamos esos valores como protección y no forzar al servo.

```
int der;//Giro a la derecha
int izq;//Giro a la izquierda
int pos=160;//Inicializamos la muñeca en el posición horizontal

void setup() {
  // put your setup code here, to run once:
  pinMode(0,INPUT);
  pinMode(2,INPUT);
  pinMode(1,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Leemos las señales procedentos del robot.
  der=digitalRead(2);
  izq=digitalRead(0);
  if (der==HIGH){
    if (pos<250){
      pos=pos+2;//Giramos de dos en dos hacia la derecha
    }
    else{
      pos=250;//Forzamos un valor máximo
    }
  }
  else if(izq==HIGH){
    if(pos>10){
      pos=pos-2;//Giramos de 2 en 2 hacia la izquierda

    }
    else{
      pos=10;//Forzamos un valor mínimo
    }
  }
  else{
```

```
      pos=pos;//Si no se activa der ni izq,
              //dejamos el servo en la misma posicion
    }
  analogWrite(1,pos);//Mandamos la posición al servo
  delay(10);//Añadimos algo de tiempo entre lecturas
}
```

Ilustración 30. Código para Arduino Gemma

Una vez el programa ha sido compilado se procederá a subirlo al Gemma. En primer lugar conectamos el Gemma al PC y veremos que el led rojo empieza a parpadear (alrededor de 25 veces), esto significa que el Gemma está “escuchando” y se le puede cargar el programa. En caso de que el led no parpadeara se pulsaría el botón que está entre el led rojo y verde. Una vez pase este periodo de escucha, el arduino pasará a ejecutar el código, siendo necesario repetir los pasos para cargar otro programa.

6 KIWIBOT BASIC SHIELD

Se ha optado por usar el escudo de KiwiBot porque es una placa muy económica con gran potencial, muchos componentes y fiable para realizar una infinidad de aplicaciones distintas. Además en la web de la asociación se puede encontrar con facilidad toda la documentación necesaria de los componentes que forman la placa, así como un foro donde encontrar numerosas respuestas a cualquier duda que pueda surgir.

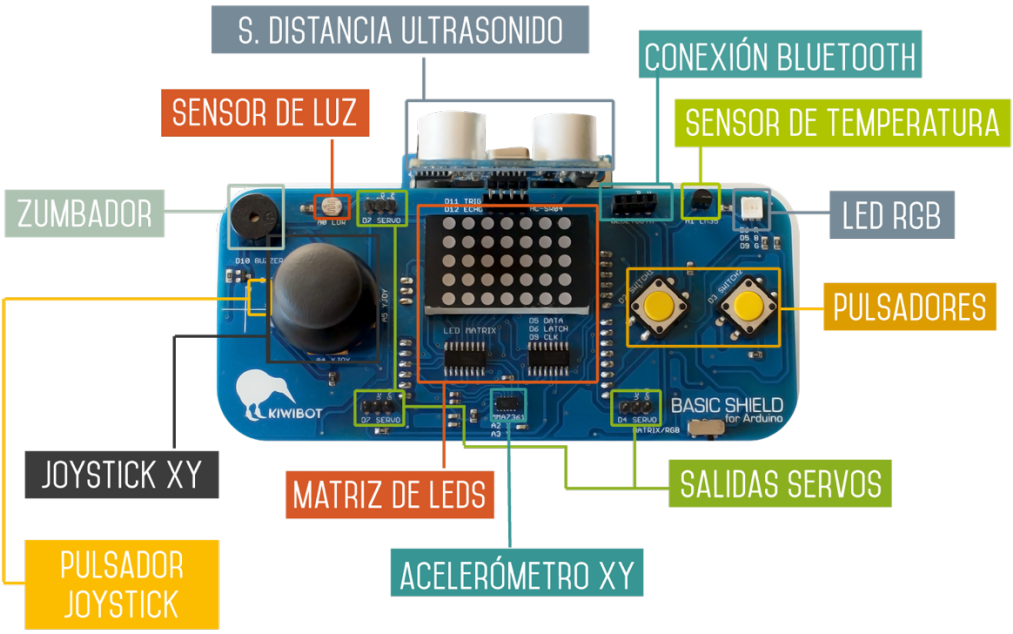


Ilustración 31. KiwiBot Shield

Como es de suponer en las imágenes la placa se encaja en el PinOut del Arduino UNO ocupando la totalidad de sus pines. El escudo de KiwiBot consta de muchos componentes, señalados arriba en la figura X.Y. A continuación se citan de manera más detallada los que se van a usar en nuestro proyecto:

Entradas:

- Joystick de dos ejes con pulsador central. Modelo Polyshine FJN10K-S1B10kD08. Pines A4/A5/D13
- Acelerómetro de dos ejes. Modelo MMA7361. Pines A2/A3
- Dos botones. Modelo B3F-4005 Pines D2/D3
- Conexión para Bluetooth. Modelo HC-05. Pines D0/D1

Salidas:

- Matriz de Leds. Modelo 703-0195. Pines D5/D6/D9

Para administrar la pantalla de leds, es necesario también el uso de un registro de desplazamiento para especificar el led que queremos controlar, en el caso del escudo KiwiBot se usa el modelo SN74HC595. El uso del conjunto de estos dispositivos sería algo tedioso, pero en el foro de la Asociación han creado una librería, llamada Matriz.h, para Arduino con funciones específicas para el manejo de la pantalla, esta librería debe añadirse a nuestro IDE. Las funciones usadas se exponen a continuación:

- Matriz.printPixel (fila,columna): enciende un led en la fila y columna seleccionada. Las filas y columnas se enumeran de manera ascendente desde la esquina superior izquierda.

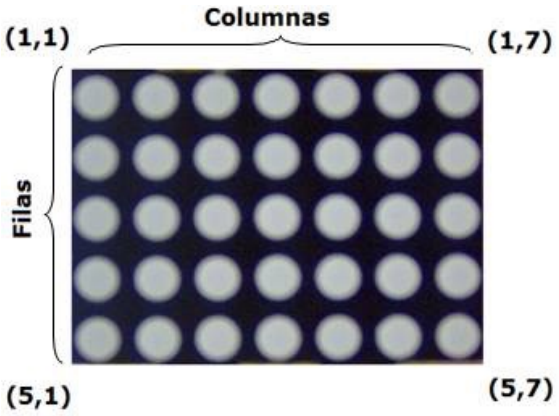


Ilustración 32. Pantalla Led de KiwiBot

- Matriz.printLinea (filaIni, ColIni, FilaFin, ColFin): Enciende una línea de leds desde el punto (FilaIni,ColIni) hasta el punto (FilFin,ColFin).

7 ARDUINO UNO

Para administrar todos los componentes usados en la placa Kiwi y mandarlos por bluetooth al robot se usa el microcontrolador Arduino UNO. Se va a realizar en primer lugar una breve introducción sobre este microcontrolador para después abordar de forma detallada la programación.

Arduino es una tarjeta controladora con una serie de entradas y salidas programadas para que desempeñen la función que se quiera. Los elementos que componen esta placa son:

Ilustración 33. Arduino UNO

Arduino se puede alimentar mediante su puerto USB, que proporciona 5V, o mediante el Jack de alimentación, que se recomienda que esté entre 7 y 12V. Los pines de alimentación son para alimentar los circuitos o placas que se conecten, en nuestro caso la KiwiBot Basic Shield:

- 3.3V: Proporciona una tensión de 3.3 V y una intensidad máxima de 50 mA.
- 5V: Proporciona 5 V y una corriente máxima de 300 mA.
- GND: Toma de tierra (0V).
- Vin: Proporciona la tensión máxima a la que está alimentada la placa.

Los pines pueden ser usados como entradas o salidas, en función de cómo se configuren pueden ser:

- Salidas digitales: Pueden el valor de LOW (0V) o HIGH (5V).
- Entradas digitales: Se interpretará como LOW un valor entre 0 y 2 V y como HIGH un valor entre 3 y 5 V.
- Salidas analógicas: Son generadas mediante PWM (Power-Width Modulation), los valores de salida van desde 0 hasta 5V con una precisión de 8 bits, es decir, 255 valores.
- Entradas analógicas: Los valores van de 0 a 5V en un rango de 0 a 1023 (10 bits de precisión).

La intensidad máxima de todos los pines es de 40 mA. En la siguiente figura puede verse todo el PinOut de Arduino UNO de manera más detallada:

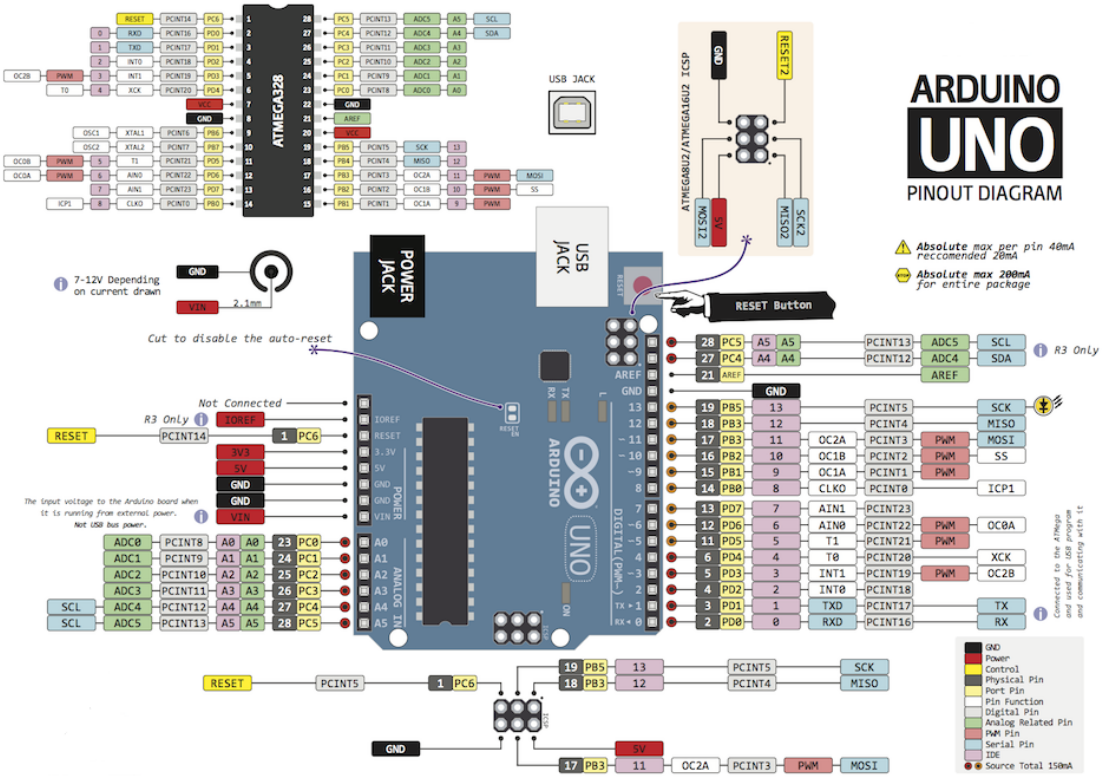


Ilustración 34. Esquemático Arduino UNO

Al conjunto de Arduino y el escudo Kiwi se le ha acoplado un portapilas (9 V) para la alimentación para no tenerlo todo el rato conectado al ordenador y darle así más libertad de movimiento.

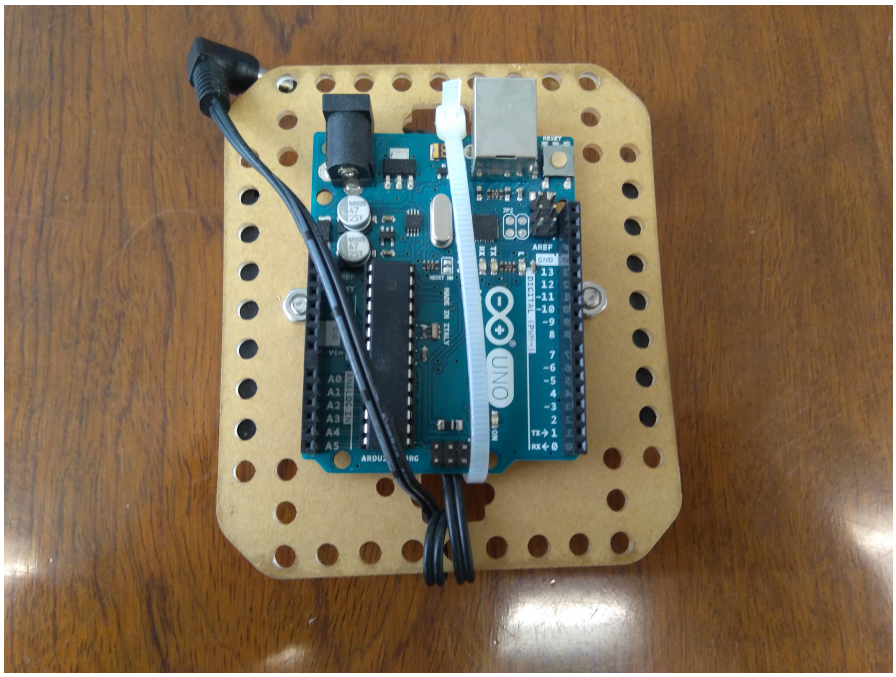


Ilustración 35.Arduino 1 con portapilas

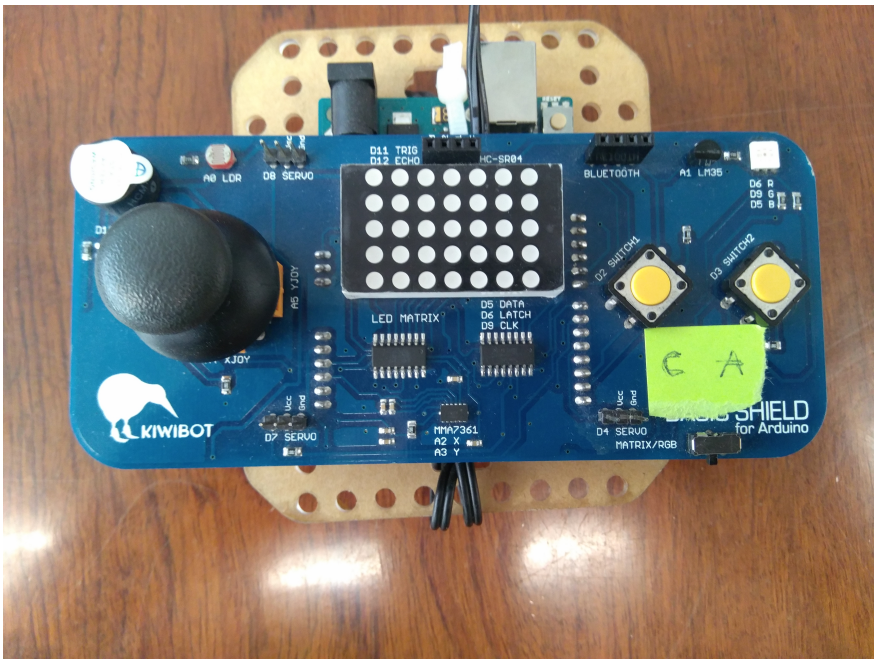


Ilustración 36. Arduino 1 con Kiwi

Ilustración 37. Arduino con portapilas trasera

La idea del programa de este Arduino consiste en asignar un comando a cada acción que se realice con el KiwiBot y pasarlo por el puerto serie a través del Bluetooth al Arduino del robot para allí realizar la actuación correspondiente. El funcionamiento del conjunto del Arduino con KiwiBot consiste en controlar con el Joystick el desplazamiento del robot a través de las dos orugas del robot, con los dos botones la apertura y el cierre de la garra y por último, pulsando el Joystick y accionando su botón central, se activaría el acelerómetro que controlaría el giro de la garra y el desplazamiento de la grúa.

Antes de asignar los comandos se tiene que definir a cada comando los rangos del acelerómetro y del Joystick que corresponde a cada movimiento, es decir los valores que toman nuestros componentes en las posiciones deseadas. Para ello se han creado dos Sketch para Arduino que nos muestra por el puerto serie el valor de los ejes X/Y cada 0,1 segundos.

En el caso del acelerómetro:

```
int xAce = A2;
int yAce = A3;
int xPosAce = 0;//Inicializamos las posiciones a Cero
int yPosAce = 0;

void setup() {
  // inicializar las comunicaciones en serie a 9600 bps:
  Serial.begin(9600);

  pinMode(xAce, INPUT);
  pinMode(yAce, INPUT);
}

void loop() {
  xPosAce = analogRead(xAce);
  yPosAce = analogRead(yAce);

  Serial.print("X:");
  Serial.print(xPosAce);
  Serial.print("\t");
  Serial.print("Y: ");
  Serial.println(yPosAce);
  delay(100); // añadir un poco de tiempo entre lecturas
}
```

Ilustración 38. Código para valores del Acelerómetro

Los valores para los ejes x e y varían desde 0 a en torno a 1023. Para las posiciones del acelerómetro se han elegido los siguiente valores de los ejes:

- Arriba:

  X=330

  Y=500
- Izquierda:

  X=490

  Y=350
- Abajo:

  X=330

  Y=190
- Derecha:

  X=170

  Y=350

De forma análoga al acelerómetro, se ha procedido con el Joystick, el código usado es el siguiente:

```
int xPin = A4;
int yPin = A5;
int buttonPin = 13;

int xPosition = 0;//Inicializamos las posiciones a Cero
int yPosition = 0;
int buttonState = 0;

void setup() {
  // inicializar las comunicaciones en serie a 9600 bps:
  Serial.begin(9600);
  pinMode(xPin, INPUT);
  pinMode(yPin, INPUT);
  pinMode(buttonPin, INPUT_PULLUP);//activar resistencia pull-up en
}

void loop() {
  xPosition = analogRead(xPin);
  yPosition = analogRead(yPin);
  buttonState = digitalRead(buttonPin);

  Serial.print("X: ");
  Serial.print(xPosition);
  Serial.print(" | Y: ");
  Serial.print(yPosition);
  Serial.print(" | Button: ");
  Serial.println(buttonState);
  delay(100); // añadir un poco de tiempo entre lecturas
}
```

Ilustración 39. Código para valores del Joystick

Los valores de los ejes X/Y varían en este caso entre 0 y 1024.

Una vez definidos los rangos que queremos para el desplazamiento del robot, se puede pasar a asignar a cada acción un comando que es el que será enviado al otro Arduino. A continuación se expone una lista con todos ellos.

- “a”: Movimiento izquierda de la muñeca.
- “b”: Movimiento derecha de la muñeca.
- “c”: Control de la grúa.
- “e”: Movimiento de orugas.
- “n”: Cerrar garra.
- “o”: Abrir garra.
- “r”: Reposo.

Además de estos comandos se envían los valores del acelerómetro y el joystick. Por ejemplo el Arduino con el Kiwi mandaría para las orugas (Joystick) el siguiente paquete por puerto serie:

```
Serial.print("<");
Serial.print("e");
Serial.print(",");
Serial.print(xPosition);
Serial.print(",");
Serial.print(yPosition);
Serial.print(">");
```

Ilustración 40. Ejemplo de encapsulado para envío de datos por puerto serie

Donde:

- <,> serían los marcadores de inicio y fin del paquete.
- e: Comando para saber que se trata de los datosdel Joystick.
- Comas para separar los datos.
- xPosition e yPosition son los datos leídos por el JoyStick.

Además se ha usado la pantalla de KiwiBot, a través de una función llamada Matrix, para indicar en todo momento que sistema estamos controlando si las orugas, marcado con la letra “O”, o la grúa que se marca con una “G”.

```
int Matrix (int grua)
{
  if(grua==HIGH)
  {
//Letra G:
  matriz.printLine(1, 2, 1, 5);
  matriz.printLine(1, 2, 5, 2);
  matriz.printLine(5, 2, 5, 5);
  matriz.printLine(5, 5, 3, 5);
  matriz.printPixel(3,4);
  matriz.printPixel(3,5);
  matriz.printPixel(2,2);
  }
  else
  {
//Letra O: Arreglar parpadeo de los leds.
  matriz.printLine(1, 2, 1, 5);
  matriz.printLine(1, 2, 5, 2);
  matriz.printLine(5, 2, 5, 5);
  matriz.printLine(5, 5, 1, 5);
  matriz.printPixel(2,5);
  matriz.printPixel(3,5);
  matriz.printPixel(2,2);
  }
}
```

Ilustración 41. Función de gestión de la matriz

Teóricamente, las letras podrían haberse realizado solo con la función llamada “matriz.printLine()”, pero se quedaban algunos leds apagados, se preguntó en el foro de la página web y no supieron el motivo ni la posible solución, por lo que se optó por encender independiente los leds apagados con la función “matriz.printPixel()”, de esta manera el problema se solucionó aunque no sea la forma más óptima.

ANEXO A: SEÑALES PWM

Arduino UNO no tiene salidas analógicas, pero puede simularlas a través de PWM (Pulse-Width Modulation), esto es, logra producir el efecto de una señal analógica a partir de la variación de la frecuencia y ciclo de trabajo de una señal digital.

El ciclo de trabajo describe la cantidad de tiempo que la señal está activa (nivel alto) como un porcentaje del tiempo total que este toma para completar un ciclo completo. La frecuencia determina como de rápido se completa un ciclo, por lo que también influye en la rapidez de cambio entre nivel alto y bajo. Este cambio realizado con la suficiente rapidez y con un cierto ciclo de trabajo se comporta como una señal analógica. Puede verse de manera más sencilla en las siguientes imágenes.

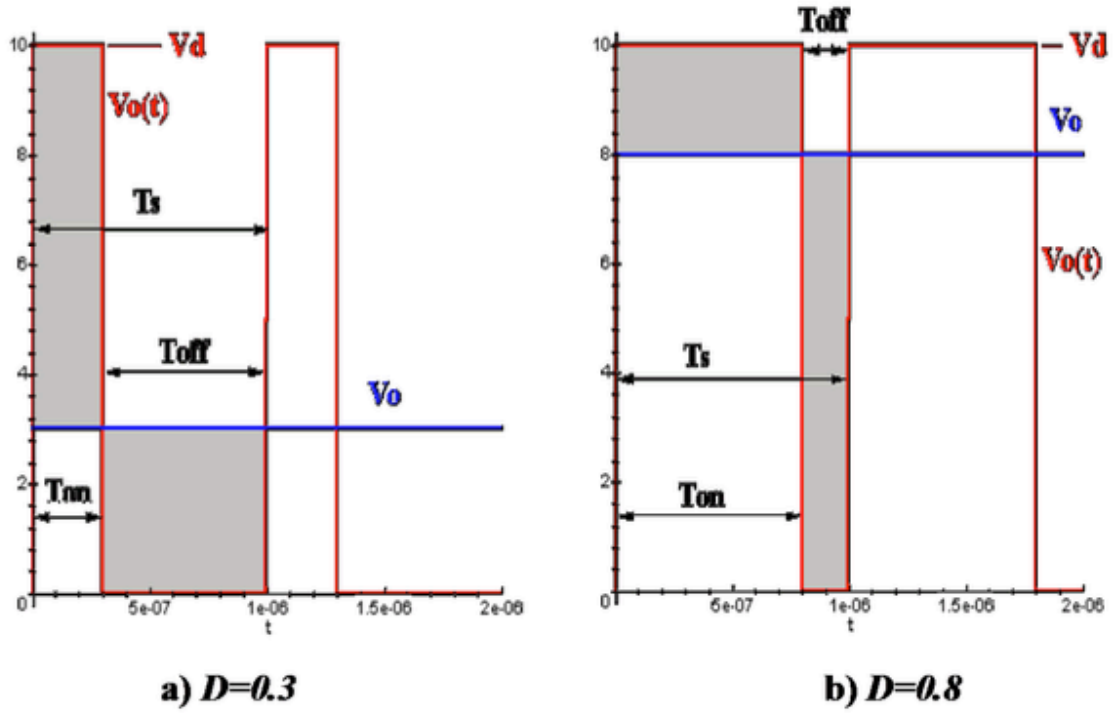


Ilustración 42. Funcionamiento de PWM

En el ejemplo se define D como el cociente entre el tiempo que está la señal a nivel alto (Ton) y el período total (Ts), variando este valor, el valor medio de la tensión de salida (Vo) es más alto o más bajo.

ANEXO B: AÑADIR LIBRERÍAS AL IDE ARDUINO

Para poder usar la placa MeOrion primero hay que instalar los drivers de la placa e incluir las librerías correspondientes en el IDE.

Las librerías son colecciones de funciones que facilitan la conexión de periféricos con Arduino. El entorno de Arduino ya incluye algunas de "fábrica". Para saber como funcionan dichas librerías hay que consultar en la web oficial de Arduino (https://www.arduino.cc/en/Reference/Libraries) o en la web donde se ha descargado la librería. Las librerías, normalmente, incluyen los siguientes archivos comprimidos en un .zip:

- Archivo .cpp (Código de C).
- Archivo .h o encabezado de C.
- Archivo Keywords.txt.
- Archivo Readme con información adicional.
- Ejemplos ya funcionales.

Los pasos a seguir para instalar la librería son:

1. Descargar la librería que se quiera instalar.
2. En el IDE de Arduino ir a Programa/Incluir librería/Añadir librería .ZIP.
3. Se abrirá una ventana donde se seleccionará la carpeta donde está el archivo .zip previamente descargado. Seleccionamos dicho archivo.
4. Volver a abrir Programas/Incluir librería y la librería instalada deberá aparecer en la parte baja del desplegable.

Para poder usar los ejemplos, en el caso de que estén incluidos, de la librería hay que reiniciar el IDE. Una vez reiniciado, estos se encuentran en Archivo/Ejemplos.

ANEXO C:REALIZACIÓN DE PCB PARA POTENCIÓMETRO Y BLUETOOTH

El kit de Makeblock contiene un módulo bluetooth HC06, pero no datasheet ni información relativa a dicho módulo, a la hora de trabajar con él se han presentado muchos problemas por lo que se optó por usar un módulo bluetooth comprado a parte, igual que el implementado en la placa KiwiBot.

Para la fácil colocación en el robot se realizó una sencilla y pequeña PCB a través de Eagle, software proporcionado por la Universidad de Sevilla. Ya se sabía usar dicha herramienta debido a haberla usado en otras asignaturas, aunque las dudas que han ido surgiendo se han resuelto gracias a este tutorial https://es.slideshare.net/antonioortegavalera/tutorial-diseo-de-pcb-con . Dicha PCB consiste en una toma RJ 25 de 6 hilos, de los cuales solo se usarán 4, y una tira de 4 pines donde irá el Bluetooth.

El esquemático y el board de la placa son los siguientes:

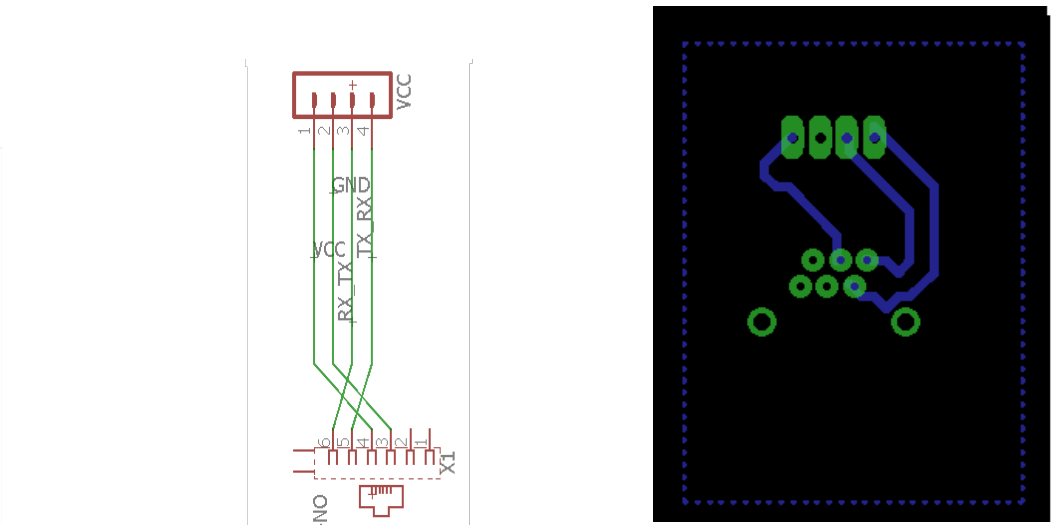


Ilustración 43.Esquemáticos del circuito para el módulo Bluetooth

Por simpleza a la hora de "enrutar" y ahorrarnos una pista, en el board se ha suprimido la pista de tierra y se ha usado un plano de masa.

La placa ha sido revelada en el taller del departamento de Ingeniería de Sistemas y Automática de la Etsi de la Universidad de Sevilla. Es un proceso sencillo que consta de los siguientes pasos:

- Introducir nuestra PCB impresa en papel vegetal y la placa en "bruto" en la insoladora durante unos dos minutos.

- Preparación del líquido revelador: Esto consiste en echar en una cubeta unas dos cucharadas de sosa cáustica por cada medio litro de agua, esta solución puede usarse varias veces. Se puede hacer mientras la placa está en la insoladora.

- Preparación del líquido atacante: Se vierte en otra cubeta 1 parte de agua oxigenada, otra de ácido clorídrico (agua fuerte) y dos partes de agua del grifo. Al contrario que la reveladora, esta solución es de un solo uso.

- Una vez haya terminado el proceso en la insoladora, introducir la placa en el líquido revelador y se espera hasta que veamos que empiezan a definirse las pistas. Cuando estén bien definidas pasamos al siguiente paso.

- A continuación se introduce la placa en la solución atacante y se mueve continuamente para mantener el líquido en movimiento durante unos dos o 3 minutos.

- Una vez esté lista la placa se limpia con agua de grifo para parar la reacción y luego con algo de acetona, para retirar resto de la película fotosensible.

- Una vez se tenga la placa lista, hay que pasar a la perforación de los pines y a la soldadura de los componentes.

Se recomienda hacer este proceso en un lugar poco luminoso, muy ventilado y con la indumentaria adecuada (guantes, gafas,bata…), ya que durante el proceso se desprenden gases nocivos para la salud. Una vez terminado todo el procedimiento, hay que reciclar las soluciones de manera adecuada.

Como resultado se ha obtenido:

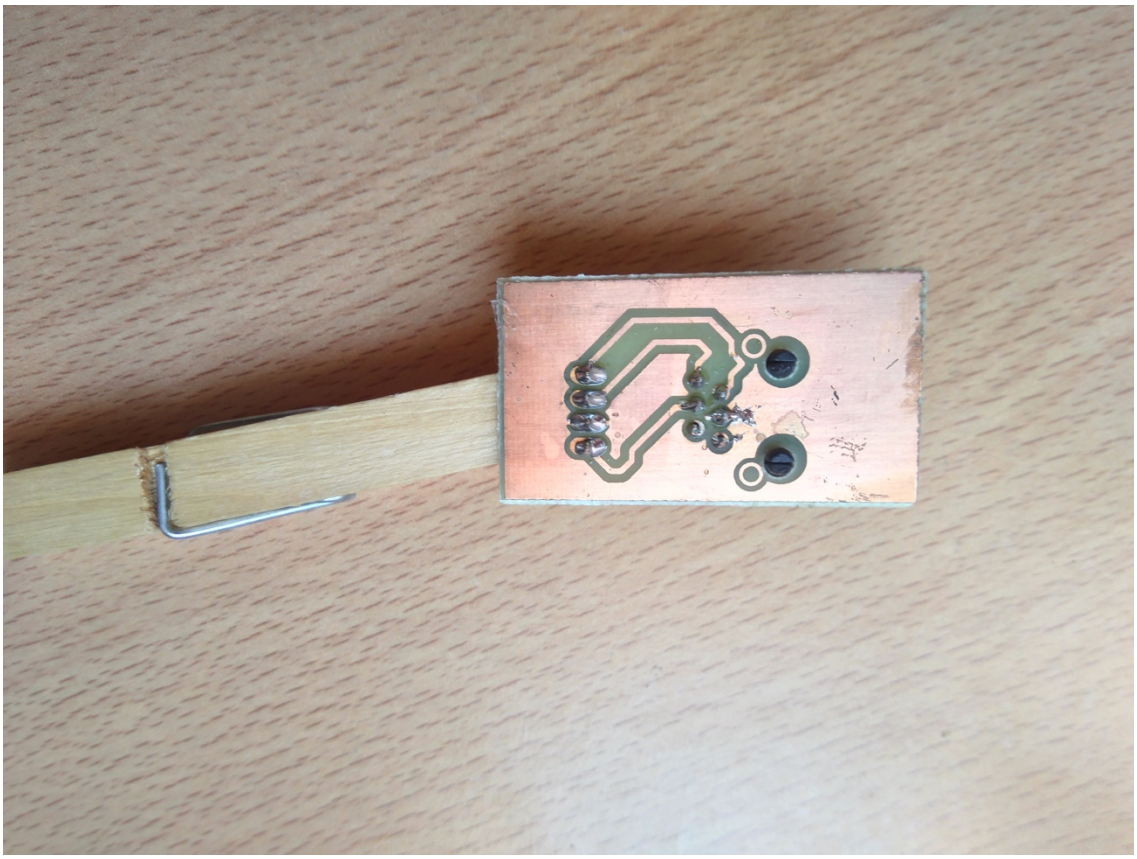


Ilustración 44. PCB para módulo Bluetooth 1

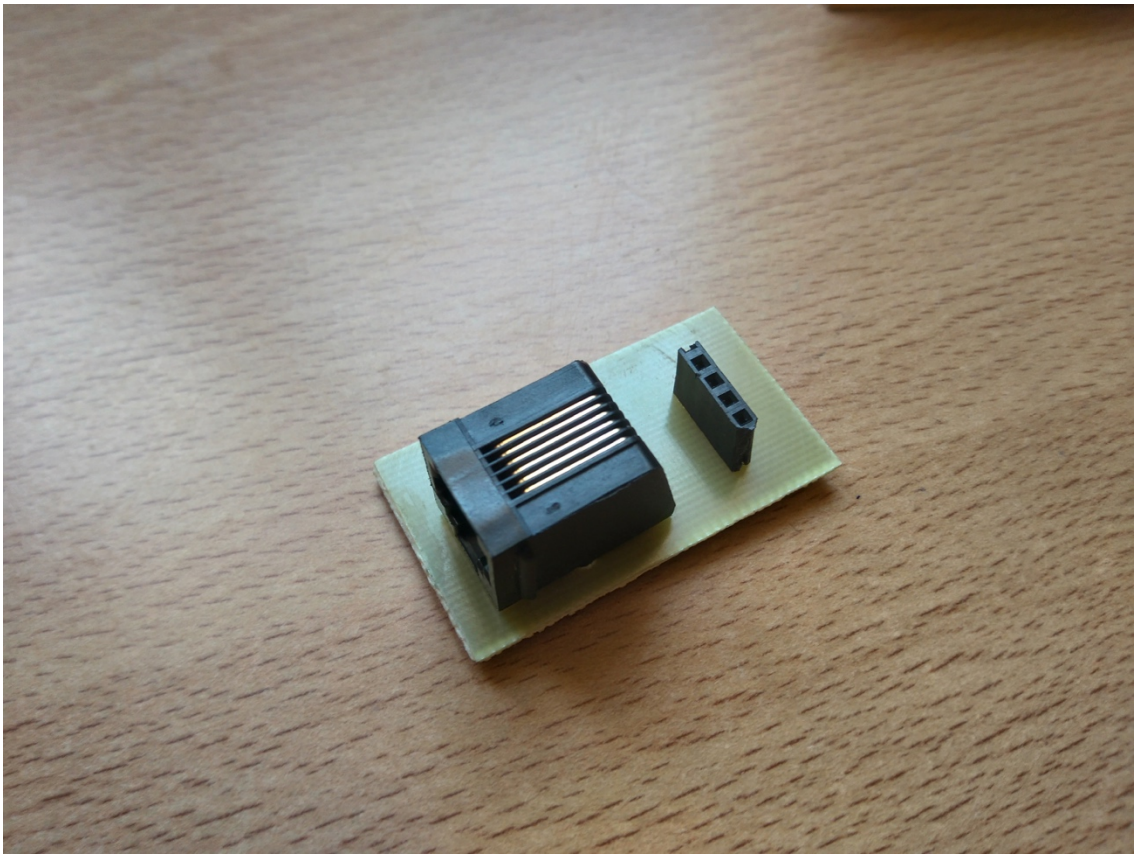


Ilustración 45. PCB para módulo Bluetooth 2

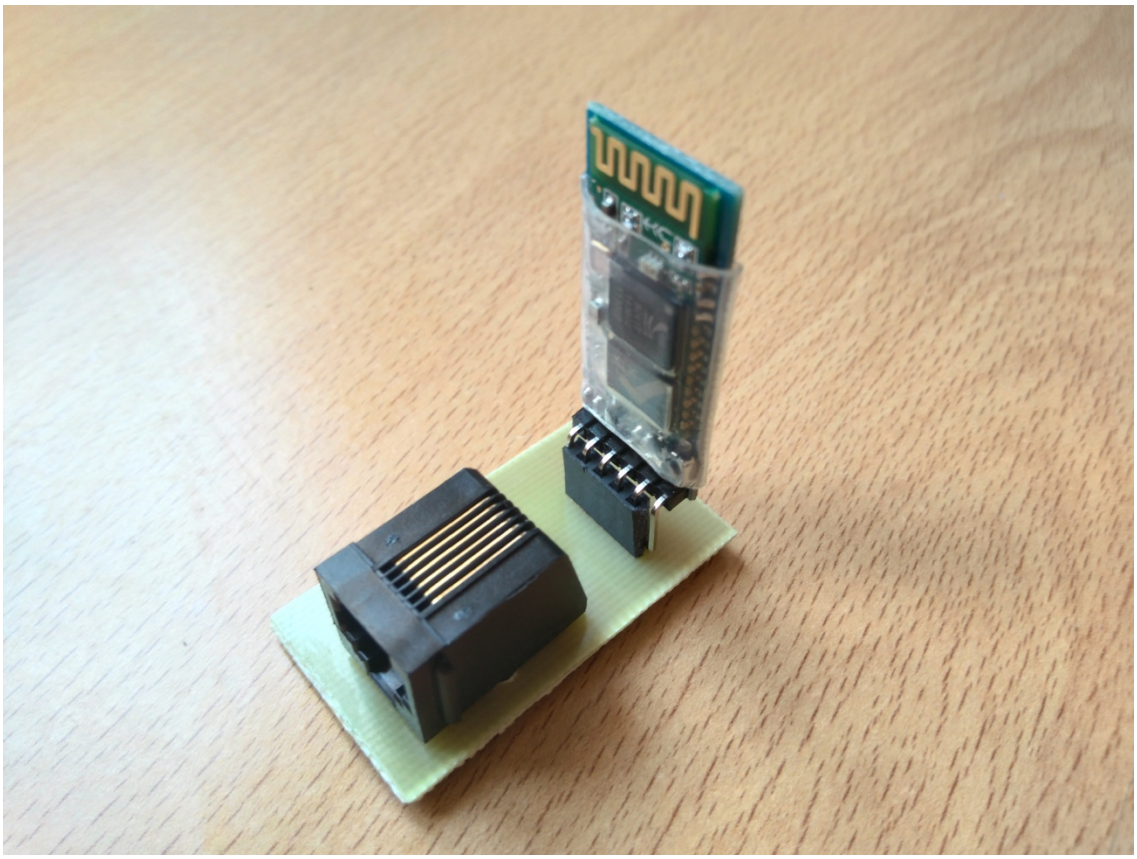


Ilustración 46. PCB para módulo Bluetooth 3

Por otro lado, se ha realizado otra "PCB" para los elementos dedicados al control automático de la grúa, está montada sobre una placa de prototipos y es algo mas rudimentaria que la primera, solo consta de un potenciómetro acoplado al eje de giro de la grúa y el cableado correspondiente. El montaje se ha realizado montando y soldando los componentes en la placa, que tiene cobre en las dos caras. La utilidad de este componente es la de saber la posición de la grúa a través del valor que tiene el potenciómetro. El circuito usado es el siguiente:

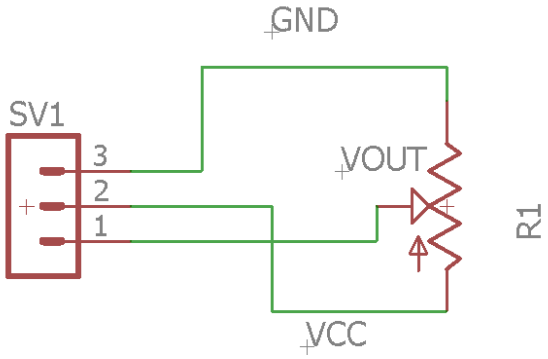


Ilustración 47. Esquemático circuito para potenciómetro

La placa de prototipos queda de la siguiente manera:

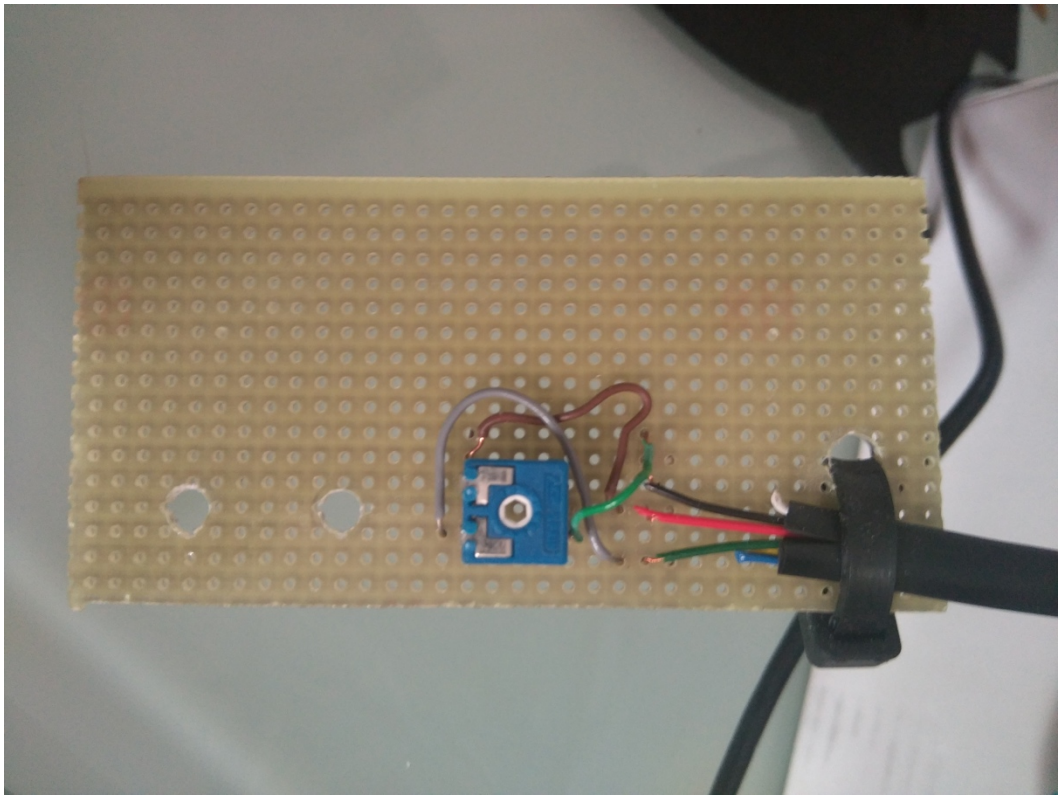


Ilustración 48. Placa de prototipos para potenciómetro 1

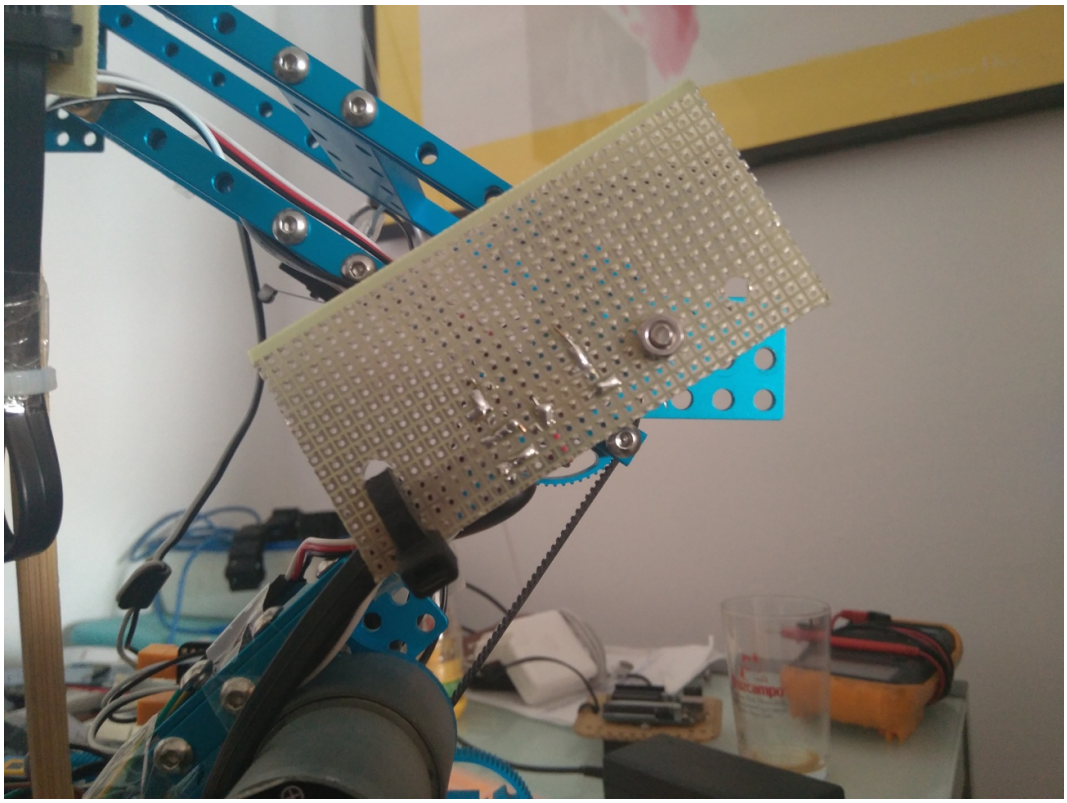


Ilustración 49. Placa de prototipos para potenciómetro 2

ANEXO D:FINALES DE CARRERA

Para aumentar la vida útil de la correa de la grúa se han acoplado dos finales de carrera para que cuando se activen, el motor se pare y la correa no resbale en los dientes de los engranajes.

Un final de carrera no es más que un interruptor que se activa cuando hace contacto con algo, tiene dos modos de funcionamiento normalmente cerrado (NC) que deja pasar siempre la corriente y se corta cuando se activa, y normalmente abierto (NO), que funciona exactamente al contrario.

En la imagen se puede ver un esquemático del dispositivo usado:

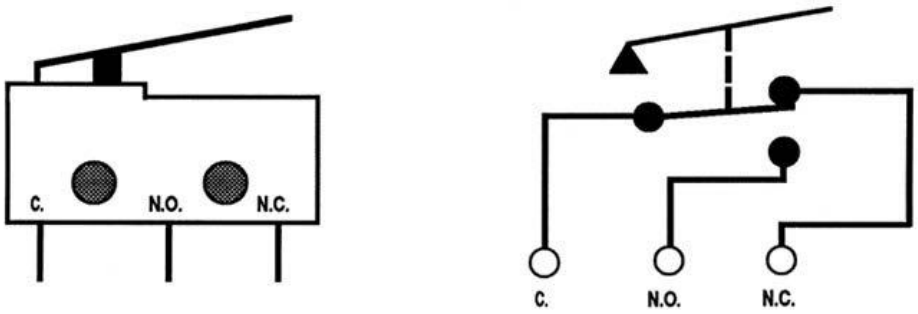


Ilustración 50. Esquemático finales de carrera

Estos dispositivo tiene muchas ventajas, como su fácil instalación y su precio, pero entre sus inconvenientes está que pueden producirse rebotes o ruidos en los contactos, para evitar esto se han usado resistencias de Pull Up o Pull Down.

Estas resistencias de Pull Up y Pull Dow son resistencias normales colocadas de manera que a la salida tengamos un estado lógico High o Low dependiendo del estado del interruptor (final de carrera en este caso). El método elegido dependerá de la funcionalidad del circuito, es decir, una resistencia Pull Up pondrá a ‘1’ el pin y un Pull Down lo pondrá a ‘0’ cuando el pin se encuentre en reposo. Las configuraciones son las siguientes:

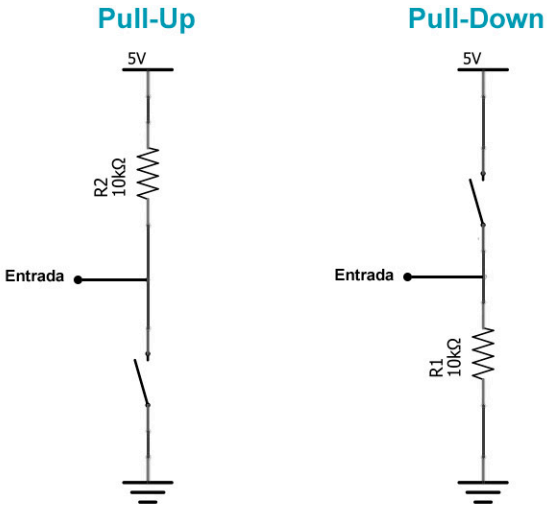


Ilustración 51. Esquemático Pull-Up/Down

Puede observase que en el caso de Pull Down, cuando se cierra el interruptor, la corriente pasará y habrá una tensión de 5V (High). Por el contrario, en Pull Up, cuando se cierra el interruptor la corriente cae a tierra y la caída de tensión es nula (Low). Las resistencias usadas suelen tener un valor de 10K.

En Arduino puede implementarse una resistencia de Pull Up en las entradas digitales mediante código, esto se hace a través de la instrucción INPUT_PULLUP cuando se inicializan los pines (función pinMode).

ANEXO E: SERIAL PLOTTER EN IDE ARDUINO

Para la obtención de las gráficas del controlador se ha usado una herramienta disponible en el IDE Arduino llamada Serial Plotter. Esta herramienta permite graficar las variables en el tiempo de manera sencilla y rápida, aunque bien es cierto que, a día de hoy, no tiene muchas funciones.

Lo único que hay que hacer para poder las variables es mostrar los datos por el puerto serie de una manera concreta, esto se ve mejor en una imagen:

```
int valor1;
int valor2;
int valor3;
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:
valor1 = random(0,100);
valor2 = random(0,100);
valor3 = random(0,100);
Serial.print(valor1);
Serial.print(“,”);
Serial.print(valor2);
Serial.print(“,”);
Serial.println(valor3);
delay(250);
}
```

Ilustración 52. Código para uso de Serial Plotter

Haciendo hincapié, se ve que las variables tienen que ir separadas por comas y que la última de estas variables tiene que llevar el retorno de carro y final de línea. Una vez se haya comprobado que el código es correcto se carga el programa al Arduino y con el cable USB conectado se clicka en: Herramientas/Serial Plotter y se abrirá una ventana con rejilla con el comportamiento temporal de las variables.

Uno de los problemas de esta herramienta es que mientras esté abierta, no podremos abrir el puerto serie (algo lógico), esto da lugar a que es muy difícil saber el valor exacto de las variables.

ANEXO F: CÓDIGOS DE PROGRAMACIÓN

En este anexo se van a ilustrar los códigos completos de todos los microcontroladores usados en el proyecto:

Arduino MeOrion

```
#include <Arduino.h>
#include <Wire.h>
#include <MeOrion.h>

//Dclaración de los motores:
MeDCMotor motor_9(9);// Motor de oruga
MeDCMotor motor_10(10);//Motor de oruga
MeDCMotor motor_1(1);//Motor de grua
MeDCMotor motor_2(2);//Motor de garra

//////////////////Variables para la interpretacion de datos:
const byte numChars=32;
char receivedChars[numChars]; //Vector de almacenamiento
char tempChars [numChars];//Vector temporal para la adquisicion de datos, ya que la
                          //funcion strtok estropearía la cadena original
/////Variables para guardar los datos:
char caracter[1]={0};
int Ejex=0;
int Ejey=0;
int orugax=0;
int orugay=0;

boolean newData=false; //Bandera para las funciones

int abajo=0;//Señal para saber que la grua ha llegado abajo
int arriba=0;//Señal para saber que la grua ha llegado arriba

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(12,INPUT);//Señal de final de carrera de grua
  pinMode(13,INPUT);//Señal de final de carrera de grua
```

```
  pinMode(2,OUTPUT);//Señal para gestionar el pwm del giro de la garra
  pinMode(8,OUTPUT);//Señal para gestionar el pwm del giro de la garra
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(2,LOW);
  digitalWrite(8,LOW);
  Recibir();
  if(newData==true){//Si hay un dato nuevo que interpretar
    strcpy(tempChars,receivedChars); //Copiamos la cadena en nuestro vector de datos recibidos
    Interpretar();//Interpreta los datos
    showParsedData();//Muestra los datos por puerto serie
    newData=false;//Indicar que ya se han interpretado los nuevos datos y se baja la bandera
  }

  switch(caracter[1]){

    case 'n'://Cerrar Garra
      motor_2.run(200);
      motor_9.run(0);//Motor Izquierdo
      motor_10.run(0);//Motor Derecho
      delay(10);

    break;

    case 'o': //Abrir Garra
      motor_2.run(-200);
      motor_9.run(0);//Motor Izquierdo
      motor_10.run(0);//Motor Derecho
      delay(10);
   break;

   case 'c':
     control();//Funcion para el control de la grua
     delay(10);
   break;

   case 'a': //Movimiento izquierda de la muñeca
     digitalWrite(8,HIGH);//Movimiento izquierda de la muñeca
     delay(10);//Damos un tiempo para enviar el siguiente 1, así se controlará mejor el movimiento
   break;

   case 'b': //Movimiento derecha de la muñeca
     digitalWrite(2,HIGH);//Movimiento izquierda de la muñeca
     delay(10);//Damos un tiempo para enviar el siguiente 1, así se controlará mejor el movimiento
   break;
// Explicacion en código KIWI para quitar este estado y limitar en "e":
   case 'r'://Estado de Reposo
     motor_9.run(0);// Motor de oruga
     motor_10.run(0);//Motor de oruga
     motor_1.run(0);//Motor de grua
     motor_2.run(0);//Motor de garra
   break;

   case 'e': //Movimiento de orugas
     orugax=map(Ejex,0,1023,0,200);
     orugay=map(Ejey,0,1023,0,200);
   //Convertimos el rango a -255/255:
     orugax=(orugax-100);
     orugay=(orugay-100);
     motor_2.run(0);
```

```
      if (orugax<25 && orugax>-25 && orugay<25 && orugay>-25){//PARADO
        motor_9.run(0);//Motor Izquierdo
        motor_10.run(0);//Motor Derecho
      }
        motor_9.run(orugay+orugax);//Motor Izquierdo
        motor_10.run(orugay-orugax);//Motor Derecho
      delay(10);
    break;
  }

}

void control (){
//Señales para el control de posición de la grúa:
  int e,sum_e=0; //Señal de error y sumatorio
  int sal;//Salida de actuación
  float Kp=4;//Constante proporcional
  float Ki=2;//Constante integral
  float Tm=0.1;//Tiempo de muestreo
  float integral;// Termino integral
  int pot;//Potenciometro

  pot=analogRead(A7);//Lectura del potenciómetro
  pot=map(pot,234,604,0,90);//Mapeamos los valores del  potenciometro.
  Ejey=map(Ejey,260,460,0,90);//Mapeamos los valores del Eje y del acelerómetro
  abajo=digitalRead(12);//Leemos el valor de los finales de carrera
  arriba=digitalRead(13);

//Calculo de la señal de control:
  e=Ejey-pot;//Calculo del error
  sum_e=e+sum_e;//Actualizamos el sumatorio del error
 integral=Ki*sum_e*Tm; //Calculamos termino integral

//Saturamos el término integral.
  if (integral>=40){
    integral=40;
  }
  if (integral<=-10){
    integral=-10;
  }

  sal=Kp*e+integral;
//Por seguridad, cortamos la señal cuando se activen los finales de carrera
  if((arriba==HIGH and sal<0) or (abajo==HIGH  and sal>0)){
    motor_1.run(0);
  }
  else{
    motor_1.run(sal);//Aplicamos al motor la salida del control
  }
    Serial.println("Datos:");
    Serial.println(Ejey,DEC);
    Serial.println(pot);
    Serial.println(e);
    Serial.println(sal,DEC);
}
void Recibir (){
  static boolean recvInProgress=false; //Bandera para saber quee estamos recibiendo datos
  static byte ndx=0;//Indice del vector
  char startMarker='<';//Marca de inicio
  char endMarker='>';//Marca de final de trama
  char rc;//Componente del vector que lee el dato del puerto serie
```

```
  char rc;//Componente del vector que lee el dato del puerto serie

  while (Serial.available()>0 && newData==false){
    rc=Serial.read();//Dato que se lee del buffer
    if (recvInProgress==true){//Bandera de lectura de datos activada
      if(rc!=endMarker){//Si es igual que ">"
        receivedChars[ndx]=rc;//Guardamos en el vector el dato
        ndx++;//Incrementamos el indice del vector
        if(ndx>=numChars){// Si llenamos el vector antes de que llegue
          ndx=numChars-1;//el caracter de fin, sobreescribimos el ultimo
        }
      }
     else{
      receivedChars[ndx]='\0';//Cerramos la cadena
      recvInProgress=false;//Bajamos la bandera de que estan llegando datos
      ndx=0;//Reiniciamos indice
      newData=true; //indicamos que hay un nuevo dato que interpretar
      }
    }
   else if (rc==startMarker){//Llega el caracter de inicio de lectura
    recvInProgress=true;//Activamos la bandera par que empiece a leer.
   }
  }
}

void Interpretar(){
  //La funcion strtok permite cortar cadenas en subcadenas mediante separadores, reemplazando estos separadores
  //por caracteres nulos \0.
  //La funcion strcpy sirve para copiar cadenas en variables
  //La funcion atoi sirve para convertir tipos de datos, en este caso de caracteres a enteros

  char*strtokIndx; // Lo usa la funcion strtok como indice

  strtokIndx=strtok(tempChars,",");//Cogemos los datos del vector recibido hasta la primera coma
  strcpy(caracter,strtokIndx);//Copia el dato en la funcion caracter

  strtokIndx=strtok(NULL,",");//Continua desde donde se dejo la ultima vez
  Ejex=atoi(strtokIndx);//convertimos el dato a entero

  strtokIndx=strtok(NULL,",");
  Ejey=atoi(strtokIndx);//convertimos el dato a entero
}

void showParsedData(){//Muestra los datos por el puerto serie
  Serial.println(caracter);
  Serial.println(orugax);
  Serial.println(orugay);
}
```

Arduino Gemma

```
int der;//Giro a la derecha
int izq;//Giro a la izquierda
int pos=160;//Inicializamos la muñeca en el posición horizontal

void setup() {
  // put your setup code here, to run once:
  pinMode(0,INPUT);
  pinMode(2,INPUT);
  pinMode(1,OUTPUT);
//  myservo.attach(1);

}

void loop() {
  // put your main code here, to run repeatedly:
  //Leemos las señales procedentos del robot.
  der=digitalRead(2);
  izq=digitalRead(0);
  if (der==HIGH){
    if (pos<250){
      pos=pos+2;//Giramos de dos en dos hacia la derecha
    }
    else{
      pos=250;//Forzamos un valor máximo
    }
  }
  else if(izq==HIGH){
    if(pos>10){
      pos=pos-2;//Giramos de 2 en 2 hacia la izquierda

    }
    else{
      pos=10;//Forzamos un valor mínimo
    }
  }
  else{
      pos=pos;//Si no se activa der ni izq,
              //dejamos el servo en la misma posicion
    }
  analogWrite(1,pos);//Mandamos la posición al servo
  delay(10);//Añadimos algo de tiempo entre lecturas
}
```

Arduino Uno

```
int grua;//Señal para alternar entre el uso de la grua o la orugas
char CerGar;//Señal para cerrar la garra
char AbrGar;//Señal para abrir la garra
//Señales para el control del acelerómetro
int xAce = A2;//Pin del eje X del acelerómetro
int yAce = A3;//Pin del eje Y del acelerómetro
//Variables para almacenar las posiciones del acelerómetro
int xPosAce = 0;
int yPosAce = 0;
//Función para incializar la matriz de ledH
#include <Matriz.h>
Matriz matriz(6, 9, 5);
//Señales para controlar el joystick
int xPin = A4;//Pin x del Joystick
int yPin = A5;//Pin y del Joystick
//Variables para guardar las posiciones del Joystick
int xPosition = 0;
int yPosition = 0;

void setup() {
  // put your setup code here, to run once:
pinMode(13,INPUT);//Boton del Joystick KIWI Probar con INPUT_PULLUP
pinMode(2,INPUT);//Switch 1 Kiwi para abrir garra
pinMode(3,INPUT);//Switch 2 Kiwi para cerrar garra
pinMode(xAce, INPUT);//Eje x del acelerometro
pinMode(yAce, INPUT);//Eje y del acelerometro
pinMode(xPin, INPUT);//Eje x del Joystick
pinMode(yPin, INPUT);//Eje y del Joystick
Serial.begin(9600);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  grua=digitalRead(13);//Leemos la entrada correspondiente al pulsador del Joystick
  Matrix(grua);
if (grua == HIGH){
//Leemos los datos del acelerómetro
  xPosAce = analogRead(xAce);
  yPosAce = analogRead(yAce);
  if      (xPosAce>400){//Izquierda
    Serial.print("<");
    Serial.print("a");
    Serial.print(",");
    Serial.print("000");
    Serial.print(",");
    Serial.print("000");
    Serial.print(">");
  }
  else if (xPosAce<275){//Derecha
    Serial.print("<");
    Serial.print("b");
    Serial.print(",");
    Serial.print("000");
    Serial.print(",");
    Serial.print("000");
    Serial.print(">");}
  else{
    Serial.print("<");
    Serial.print("c");
    Serial.print(",");
    Serial.print(xPosAce);
    Serial.print(",");
    Serial.print(yPosAce);
    Serial.print(">");
  }
  }

else
  {
//Leemos los datos del JoyStick
  xPosition = analogRead(xPin);
  yPosition = analogRead(yPin);

    if (xPosition<650 && xPosition>400 && yPosition<650 && yPosition>400){//Reposo
      Serial.print("<");
      Serial.print("r");
      Serial.print(",");
      Serial.print("000");
      Serial.print(",");
      Serial.print("000");
      Serial.print(">");
    }
    else{ //Valores X e Y del Joystick
    Serial.print("<");
    Serial.print("e");
    Serial.print(",");
    Serial.print(xPosition);
    Serial.print(",");
    Serial.print(yPosition);
    Serial.print(">");
```

```
  }
//Abrir o cerrar la garra
  CerGar=digitalRead(2);//Cerrar Garra
  AbrGar=digitalRead(3);//Abrir Garra
  if (CerGar==HIGH){
    Serial.print("<");
    Serial.print("n");
    Serial.print(",");
    Serial.print("000");
    Serial.print(",");
    Serial.print("000");
    Serial.print(">");}//Cerrar garra
  if (AbrGar==HIGH){
    Serial.print("<");
    Serial.print("o");
    Serial.print(",");
    Serial.print("000");
    Serial.print(",");
    Serial.print("000");
    Serial.print(">");}//Abrir garra
}
delay (20);
}
int Matrix (int grua)
{
  if(grua==HIGH)
  {
//Letra G:
  matriz.printLine(1, 2, 1, 5);
  matriz.printLine(1, 2, 5, 2);
  matriz.printLine(5, 2, 5, 5);
  matriz.printLine(5, 5, 3, 5);
  matriz.printPixel(3,4);
  matriz.printPixel(3,5);
  matriz.printPixel(2,2);
  }
  else
  {
//Letra O: .
  matriz.printLine(1, 2, 1, 5);
  matriz.printLine(1, 2, 5, 2);
  matriz.printLine(5, 2, 5, 5);
  matriz.printLine(5, 5, 1, 5);
  matriz.printPixel(2,5);
  matriz.printPixel(3,5);
  matriz.printPixel(2,2);
  }
}
```

ANEXO G: MONTAJE DEL ROBOT

Las ilustraciones para el montaje del robot son las siguientes:

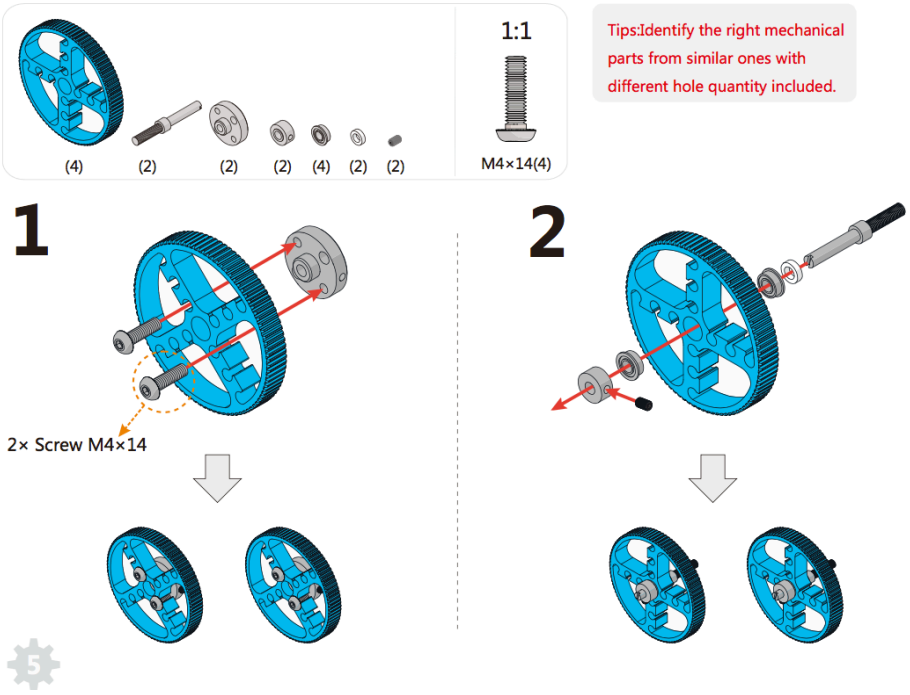


Ilustración 53. Montaje 1

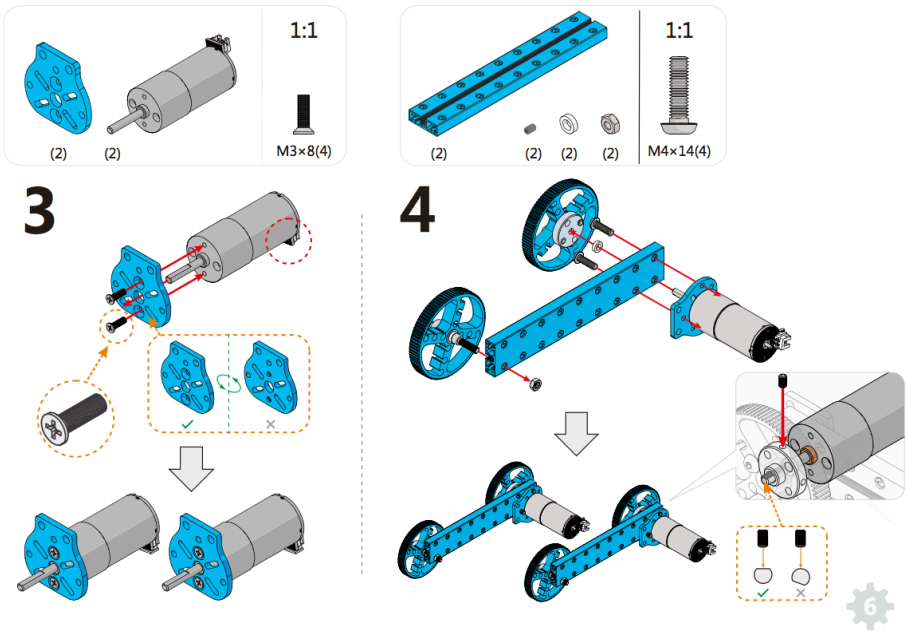


Ilustración 54. Montaje 2

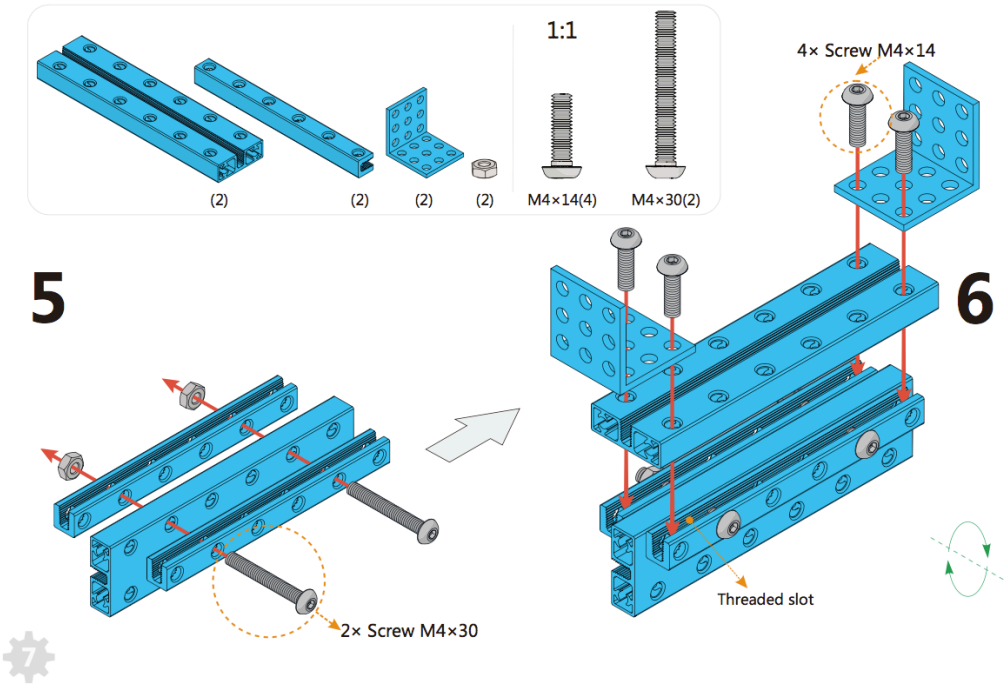


Ilustración 55. Montaje 3

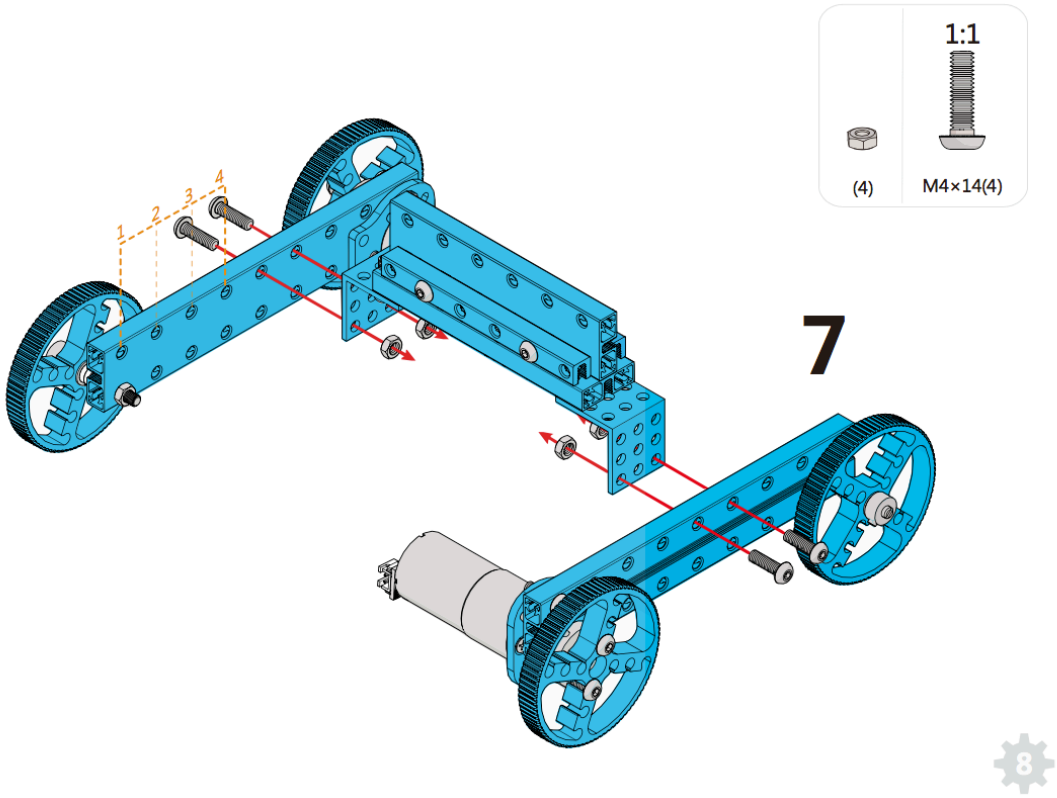


Ilustración 56. Montaje 4

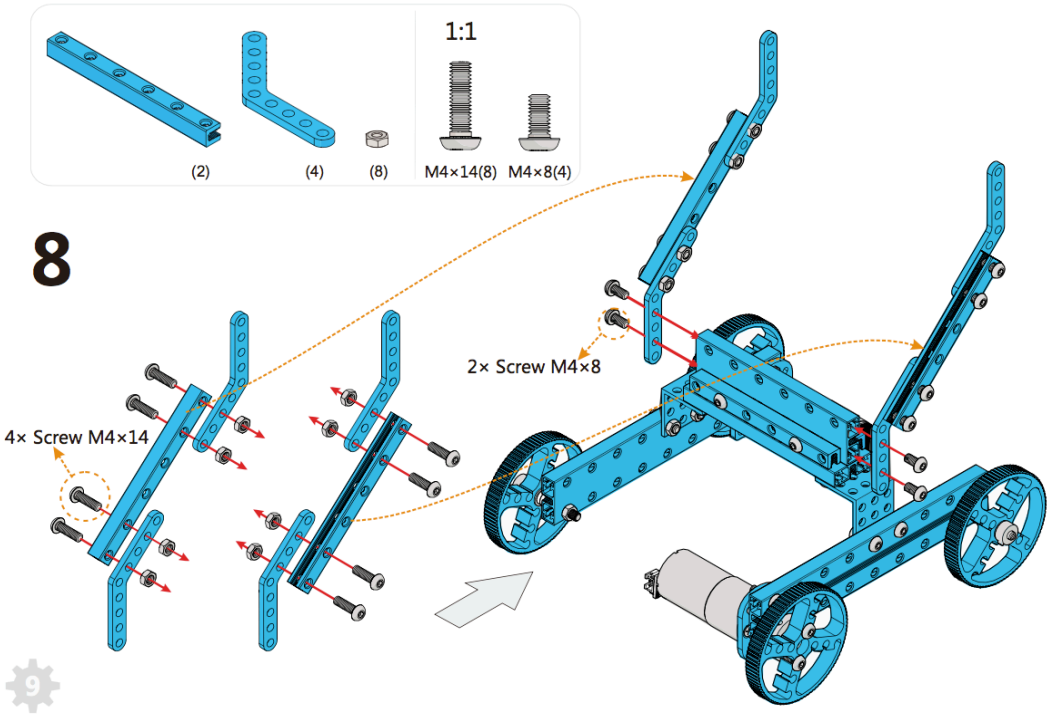


Ilustración 57. Montaje 5

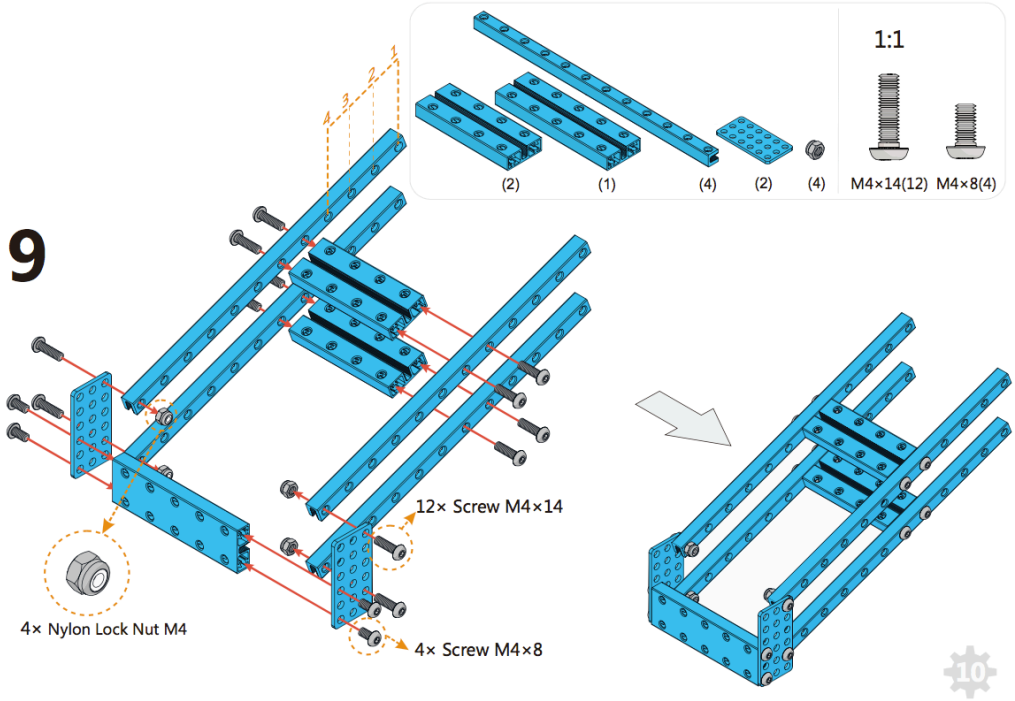


Ilustración 58. Montaje 6

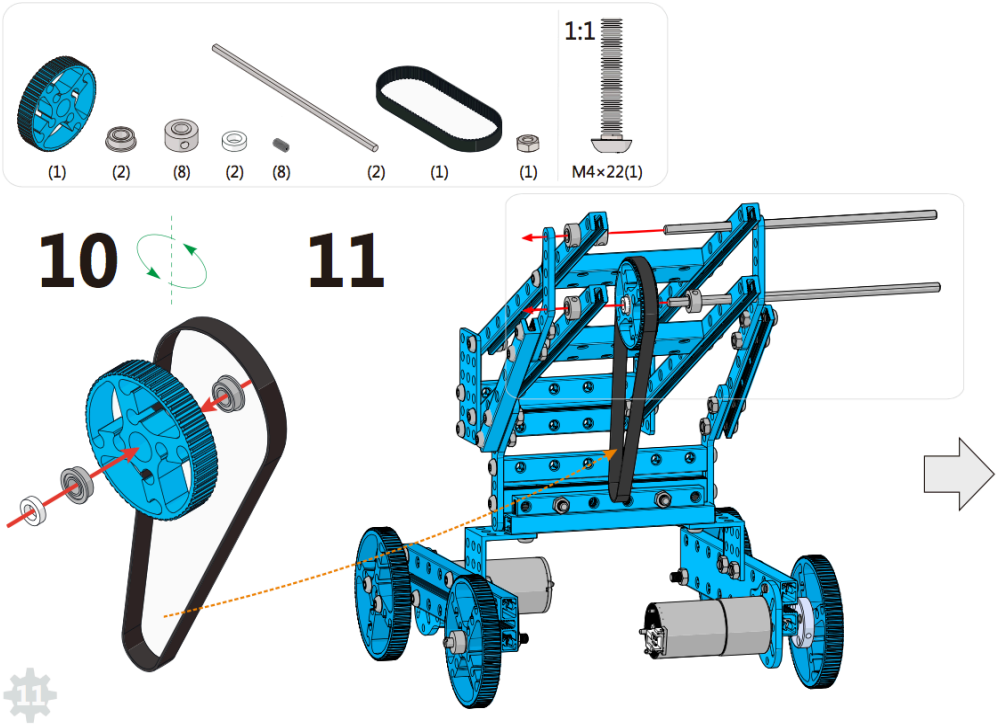


Ilustración 59. Montaje 7

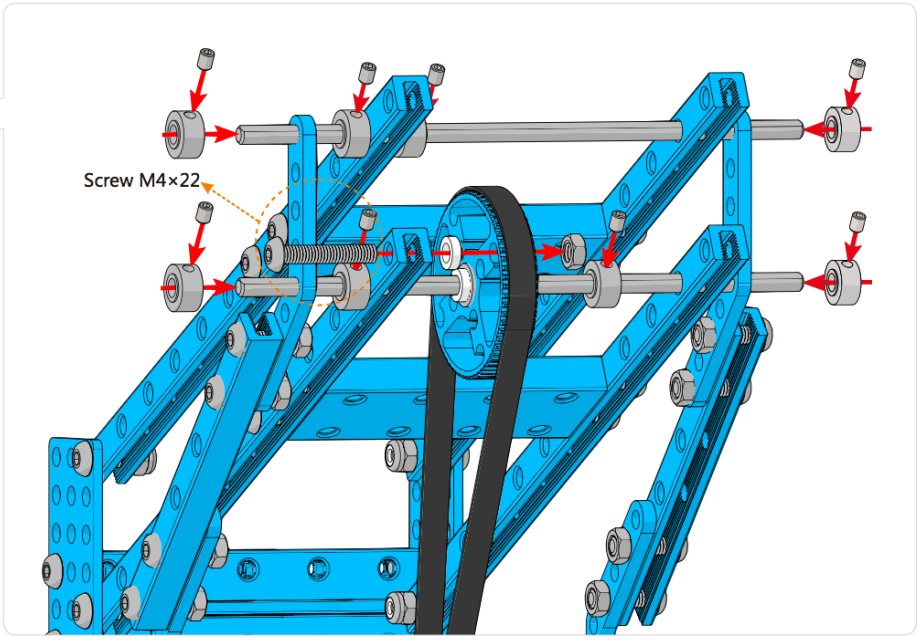


Ilustración 60. Montaje 8

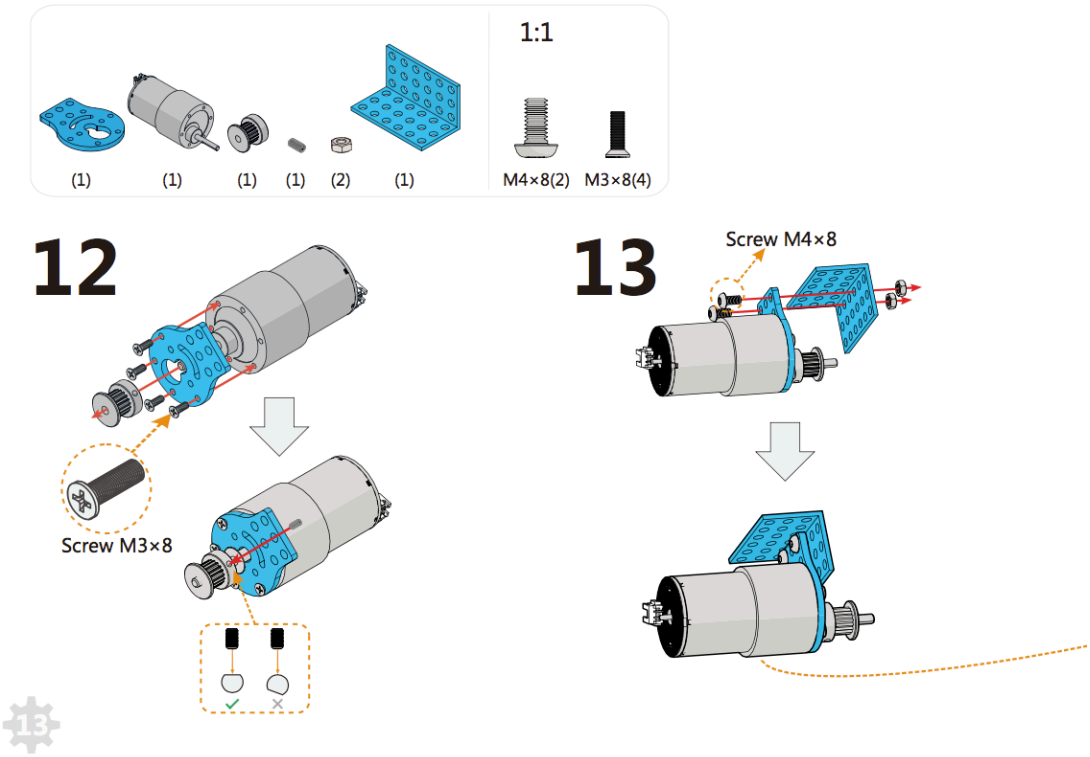


Ilustración 61. Montaje 9

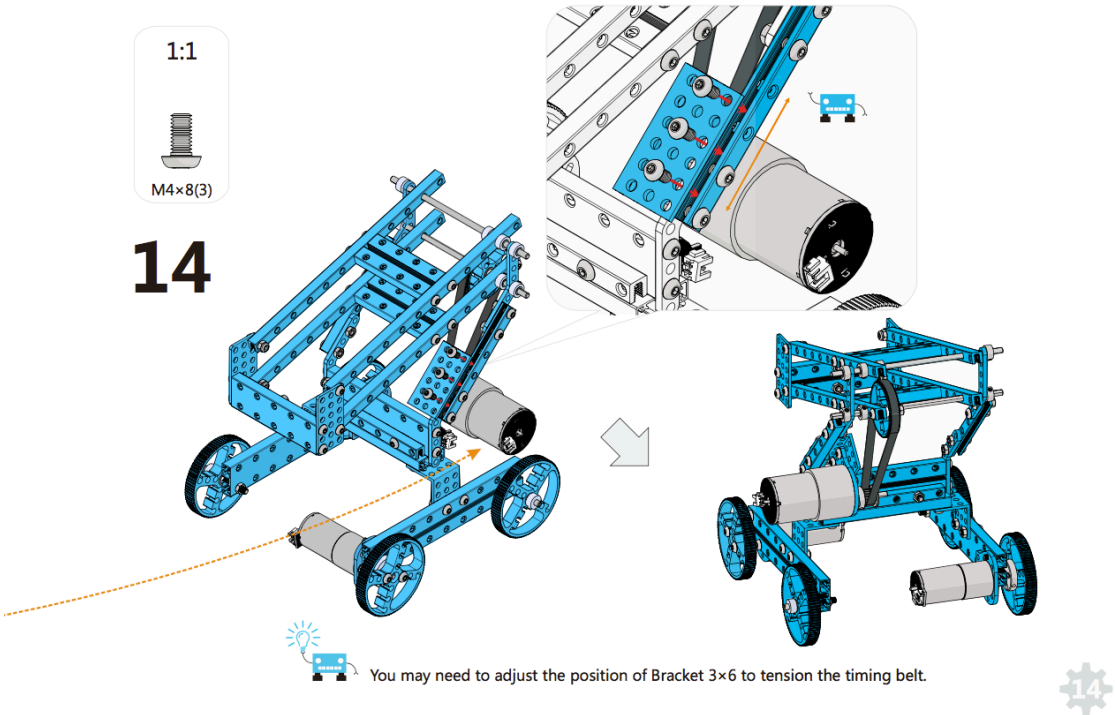


Ilustración 62. Montaje 10

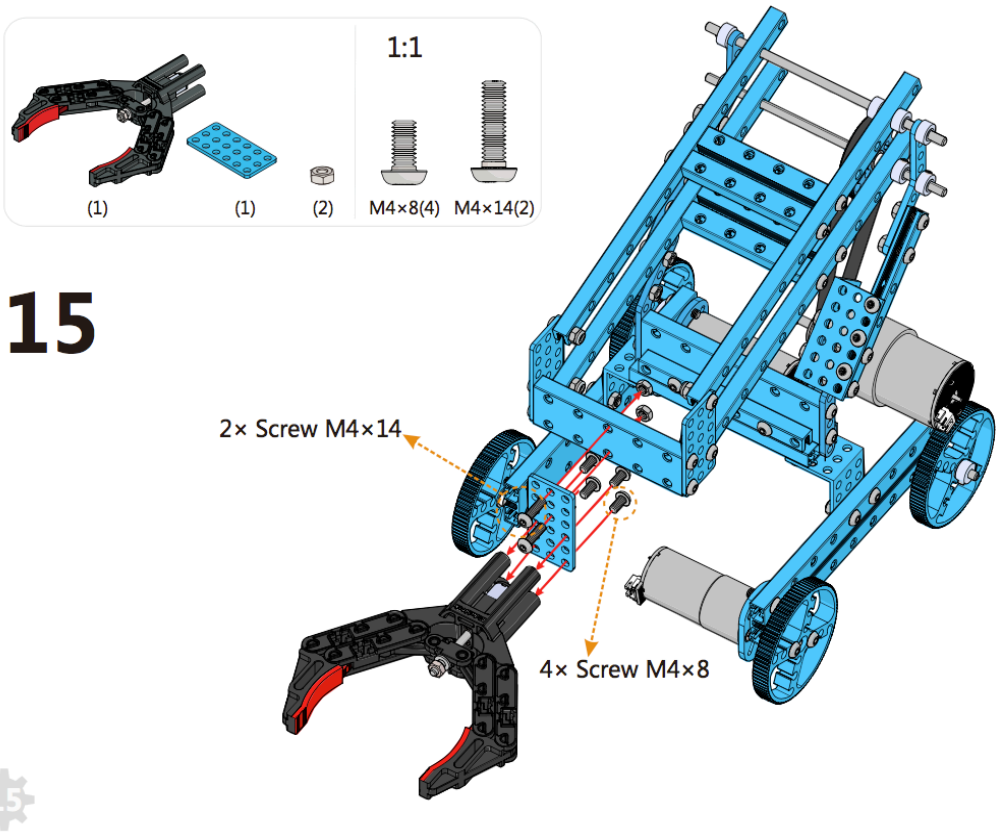


Ilustración 63. Motaje 11

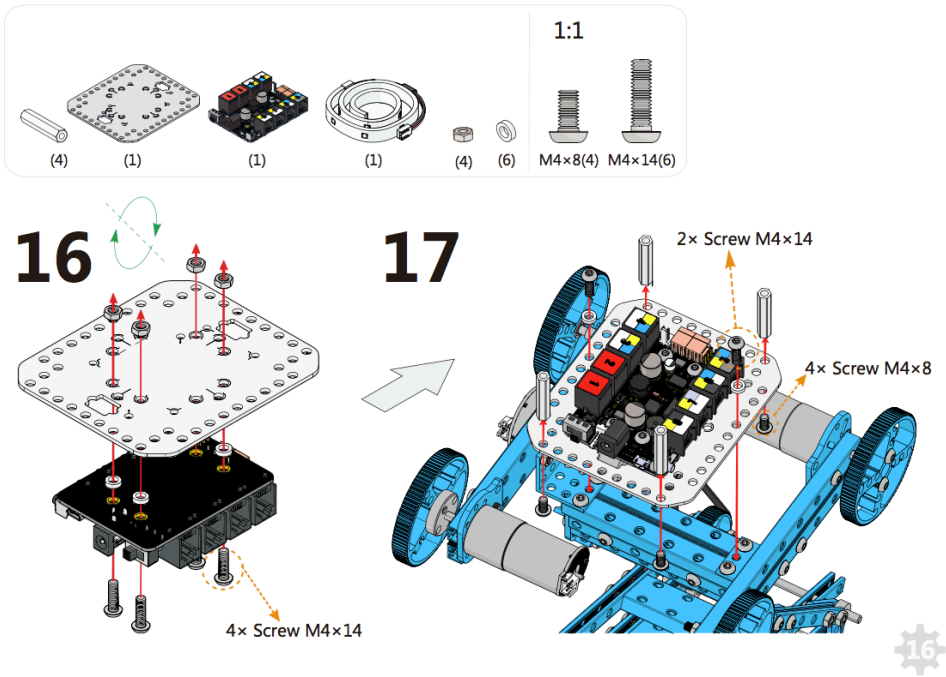


Ilustración 64. Montaje 12

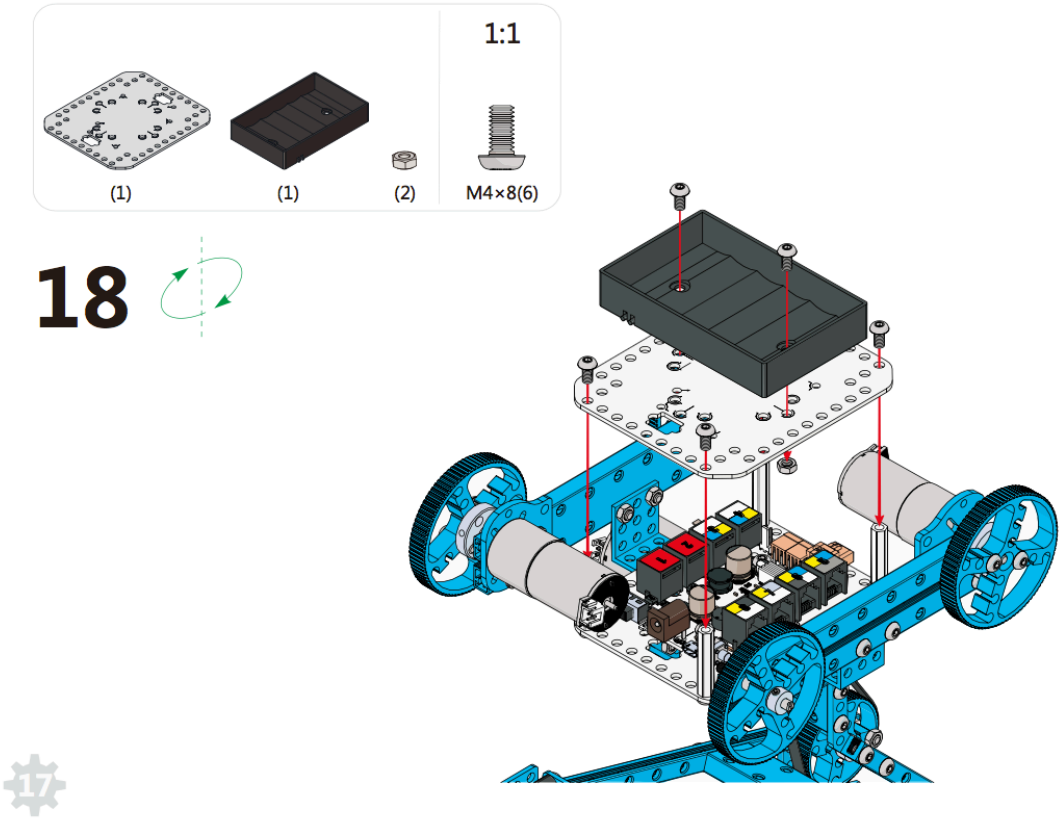


Ilustración 65. Montaje 13

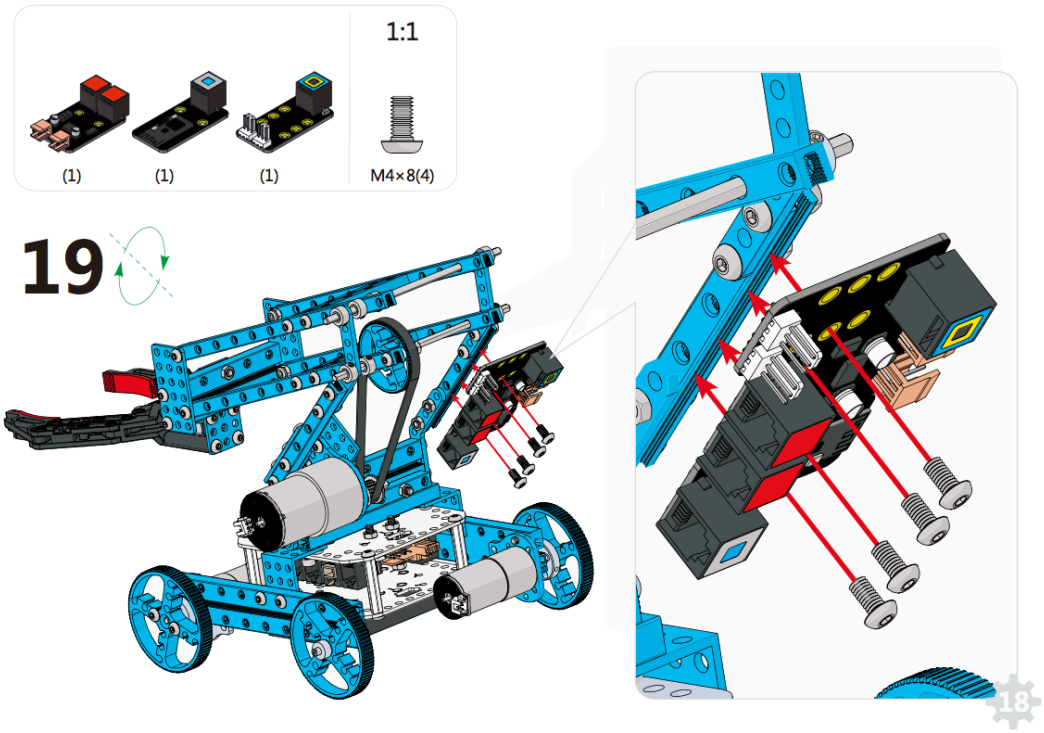


Ilustración 66. Montaje 14

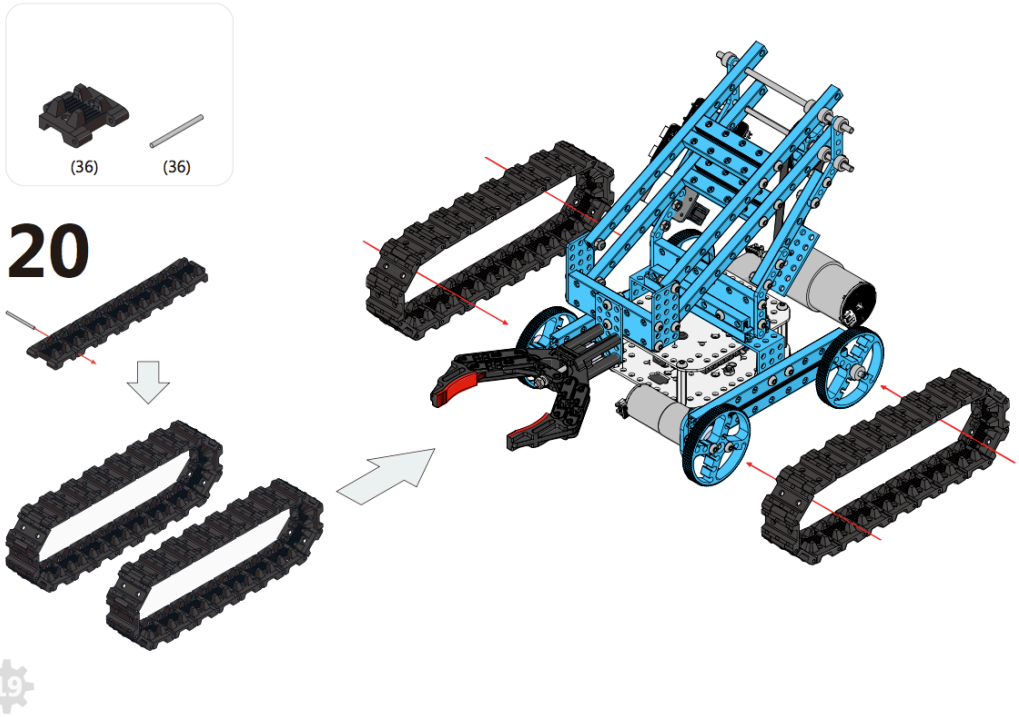


Ilustración 67. Montaje 15

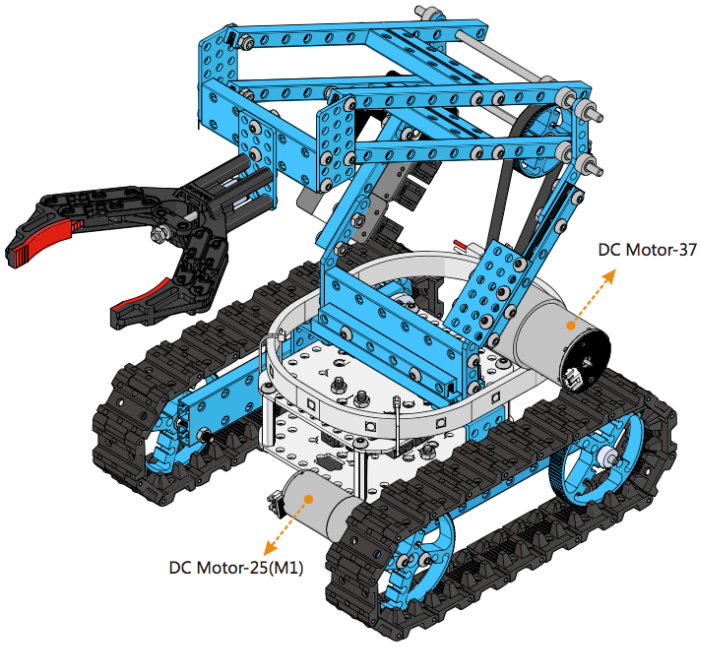


Ilustración 68. Montaje 16

REFERENCIAS

[1] https://forum.arduino.cc

[2] https://www.wikipedia.org

[3] http://learn.makeblock.com/makeblock-orion/

[4] http://www.naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html

[5] http://picfernalia.blogspot.com.es/2013/02/controloralor-pid-para-posicion-de-un.html

[6] http://www.naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html

[7] http://www.kiwibot.es

[8] http://www.askix.com/como-programar-v2-adafruit-gemma_4.html#title